Aplicación de control de pantallas de información para la FDI

Facultad de Informática
Universidad Complutense de Madrid
21 de mayo, 2019

Alumnos: Gabriel Gutiérrez Santos, Adrián Montero Torralbo y Daniel Gutiérrez Delgado.

Tutor: Marco Antonio Gómez Martín.

Aplicación de control de pantallas de información para la FDI

Facultad de Informática
Universidad Complutense de Madrid
21 de mayo, 2019

Alumnos: Gabriel Gutiérrez Santos, Adrián Montero Torralbo y Daniel Gutiérrez Delgado.

Tutor: Marco Antonio Gómez Martín.

Resumen: En la Facultad de Informática de la Universidad Complutense de Madrid, donde se desarrolla este trabajo, existen dos pantallas destinadas a ofrecer información relacionada con eventos de la facultad a todos los alumnos.
Nuestro proyecto surge de la necesidad de renovar el sistema actual que nutre a estas pantallas, el cual resulta algo obsoleto. Ahora mismo, este sistema no se puede programar para que las pantallas reproduzcan un contenido dinámico que se va generando en tiempo real.
Por esto, creamos un sistema para las pantallas de la facultad con el que se puede ver un contenido más dinámico, que además puede ser programado de manera sencilla.

Con la incorporación de nuestro proyecto los estudiantes de la Facultad de Informática podrán, mientras pasean por el vestíbulo camino de sus clases o mientras recargan sus baterías en la cafetería, ver el último tweet de @informaticaucm que se acaba de publicar o no perder ni un detalle del concurso de programación que está teniendo lugar en los laboratorios.

Palabras clave: cartelería digital, pantallas, sistema de gestión de contenido, controlador de gestión diaria, controlador de concursos, concurso de programación, sistema de prioridades, Xibo, diapositiva, reproductor.

Abstract: In Facultad de Informática of Universidad Complutense de Madrid, where this project is developed, there are two screens whose objective is offer information related to faculty to all students.
Our project arises from need to renew the current system what feeds these screens, which is some obsolete. Right now, this system can not be programmed in order to the screens display a dynamic content that is generated in real time.
For this reason, we create a system in screens of faculty in order to can watch a more dynamic content, which moreoer can be programmed in a simple way.

With the incorporation of our proyect Facultad de Informática students will can, while they are walking on the hall on way to their classes or while they are recharging their batteries on the cafeteria, watch the latest tweet of @informaticaucm that has just been published or not to miss a single detail of the programming competition that is taking place in the laboratories.

Keywords: digital signage, screens, content management system, daily management controller, contests controller, programming competition, priority system, Xibo, layout, player.

Índice

Capítulo 1. Introducción

Hoy en día cada vez es más importante estar actualizados de todos los eventos y sucesos que se están produciendo en ese preciso instante, por lo que cada vez más se intenta transmitir estos sucesos con mayor rapidez en función de su prioridad. Aplicaciones como Twitter o Facebook están constantemente proporcionando información en tiempo real, así como el formato de vídeo ya no se limita a captura y reproducción, sino que puede ser retransmitido en directo por Internet mediante *streaming*. Por ello, cada vez se valora más el poder acceder a toda la información cuanto antes de manera rápida y sencilla, y en este aspecto juegan un papel fundamental las pantallas informativas que podemos ver en muchos sitios.

Actualmente, en la Facultad de Informática de la Universidad Complutense (a partir de ahora FDI), donde surge y se desarrolla este proyecto, existen dos pantallas que muestran información de manera programada a los estudiantes, como ya pasa en muchos otros lugares públicos. Es algo muy útil, ya que permite proporcionar a los estudiantes información acerca de la facultad más allá de la metodología ya "clásica" del correo electrónico, que muchas veces es pasado por alto entre otros tantos correos; y un paso más allá, tecnológicamente hablando, de lo que sería el típico cartel en el tablón de anuncios, el cual también puede pasar desapercibido cuando hay muchos carteles.
En estas pantallas se muestran, como hemos dicho, información sobre eventos de diferente ámbito relacionados con la facultad: promoción de la semana de la Informática, de concursos de programación, charlas o conferencias, etc.

El sistema que se emplea actualmente para ello, bastante "rudimentario", se compone básicamente de una aplicación web alojada en un servidor de la facultad, la cual hace la función de gestor de contenidos (a partir de ahora *CMS*, siglas de *Content Management System* en inglés, o *Sistema de Gestión de Contenido* en castellano) y donde se recoge una lista de contenidos de distintos tipos. Para modificar el contenido a mostrar en las pantallas, se accede a esta aplicación y se eliminan, añaden o modifican los contenidos uno a uno.

Hay un ordenador, también en la facultad, ejecutando un reproductor de contenidos (Player) que recibe el contenido del CMS y lo muestra en el monitor. Las mismas imágenes que se pueden ver en este monitor son las que se reproducen en las pantallas de las que hablábamos anteriormente.

El propósito de este Trabajo Final de Grado es mejorar el funcionamiento de las pantallas de información que se encuentran en nuestra facultad de manera que puedan ofrecer información dinámicamente y en tiempo real de un modo simple y rápido.

Como hemos descrito previamente, para añadir información con el sistema actual debemos acceder al CMS y añadir el contenido de manera manual. De esta forma, información dinámica como podría ser un tweet de la cuenta oficial de la facultad resulta imposible de mostrar en las pantallas, puesto que conlleva tener a alguien añadiendo el contenido manualmente cada vez que se publicase, y eso es algo muy tedioso, anticuado, y nada práctico. La única opción existente para hacerlo posible, con el sistema actual, sería añadir un script que estuviera todo el rato ejecutándose y observarse mediante encuesta si se publica un tweet.

Partiendo de esta base, estas pantallas deberían ser capaces de estar mostrando un contenido "habitual" y, en cierto momento, pausar esa reproducción para mostrar un tweet de la cuenta oficial de la FDI[1] [2], o incluso interrumpir su programación para informar de la cancelación de las clases de manera extraordinaria, como podría ser, por una importante nevada.

Con todo esto en mente, y combinando las capacidades deseadas para nuestro proyecto, decidimos concretar una prueba de concepto sobre la cual poder demostrar el potencial de nuestro sistema: ser capaces de controlar y ofrecer en las pantallas la información en tiempo real relativa a un concurso de programación que se realice en la facultad.

[1] Web de la Facultad de Informática: https://informatica.ucm.es/

[2] Twitter de la Facultad de Informática: https://twitter.com/informaticaucm

Por ello en este trabajo se propone utilizar el CMS de la empresa Xibo Signage LTD y gestionarlo a distancia por aplicaciones externas elaboradas por nosotros que se encargarán de añadir la parte dinámica. Estas aplicaciones son:

- Controlador de gestión diaria:
  Aplicación que añadirá el contenido al CMS mencionado anteriormente para después reproducirlo en las pantallas.

- Controlador de concursos:
  Esta aplicación estará escuchando los cambios y eventos producidos durante el concurso. En función al tipo de cambios que reciba mostrará información relevante actualizada del concurso.

Este documento se estructura como se indica a continuación. Tras el capítulo con esta misma introducción en inglés, el siguiente capítulo trata del estado actual del sistema de pantallas, así como las diferentes posibles opciones para integrar un sistema renovado y funcional. Después, el capítulo 4 describe la herramienta elegida para desarrollar nuestro proyecto. Los capítulos 5 y 6 explican en profundidad las distintas aplicaciones que hemos desarrollado para llevar a cabo el objetivo del proyecto. Y, por último, el capítulo 7 trata las conclusiones obtenidas tras la finalización de dicho proyecto.

Capítulo 2. Introduction

Nowadays is increasingly more important to be updated of all events and occurrences what are taking place in that precise momento, so more and more it is trying to transmit these events more quickly according to their priority. Applications like Twitter or Facebook are constantly providing real time information, and video format is no longer only to be captured and playbacked, as it can be live broadcasted over Internet trhough streaming. For these reasons, it is increasingly valued to be able to access all information as soon as possible in a quickly and simple way, and about it informative screens that we can see in many places play fundamental role.

Currently, in Facultad de Informática of Universidad Complutense (from now FDI), where this project arises and is developed, there are two screens that displays information in a programmed way to students, as it happens yet in many other public places. It is something very useful, because it allows to provide students with information about the faculty beyond the "classic" email methodology, which is often overlooked between many other emails; and a step beyond, technollogically speaking, of what would be the typical poster on bulletin board, which can also be overlooked when there are many posters.
These screens displays, like we have said, information about events of different scope related with faculty: promotion of Computer Science Week, of programming competitions, seminars or conferences, etc.

System that is currently used for this, quite "rudimentary", is basically composed by a web application hosted over a faculty server, which acts as a content manager (from now CMS) and where a list of contents of different types is collected. In order to modify the content to be displayed on screens, this application is accessed and contents are deleted, added or edited one by one.

There is a PC, in faculty too, running a content player what receives content from CMS and displays it on the PC screen. Same images what can be watched in this PC screen are those that are displayed on the screens we talked about previously.

The purpose of this Final Degree Project is to improve the functioning of the information screens what are located in our faculty so that they can offer dynamic information and real time information in a simple and fast way.

As we have previously described, in order to add information with current system we must access to CMS and add manually content. In this way, dynamic information like it could be a tweet from the official faculty account is imposible to display on screens, because it involves to have somebody adding manually the content each time it is published, and that is something very tedious, outdated and not practical. The only existing option to make it possible, with the current system, would be to add a script what would be uninterrupted running and polling all time if a tweet has been published.

From this base, these screens would be able to are displaying an "usual" content and, at some point, pause that playback in order to display a tweet of the official FDI account, or even interrupt its programming to report cancellation of classes in an extraordinary way, as it could be, due to an important snowfall.

With all this in mind, and mixing the desired capabilities for our project, we decide to finalize a proof of concept on which to specify the potential of our system: to be able to control and offer on screens real time information related to a programming competition that takes place in the faculty.

Therefore in this project we propose to use the CMS of Xibo Signage LTD company and manage it remotely by external applications developed by us that will be responsible for adding dynamic part. These applications are:

- Daily management controller:
  Application that will add content to the previosly mentioned CMS and then display it on screens.

- Contests controller:
  This application will be listening changes and events happened during the contest. Depending on changes type what it receives, it will display updated relevant information about the contest.

This document is structured as indicated below. After this chapter, next chapter deals with the current state of screens system, and the different possible options to integrate a renewed and functional system. After this, chapter 4 describes the chosen tool in order to develop our project. Chapters 5 and 6 explain in Depth differents applications what we have developed in order to reach our project objective. And, finally, chapter 7 deals with conclusions obtained after the completion of this project.

Capítulo 3. Estado del arte

En la Facultad de Informática existen actualmente dos pantallas que muestran información a los alumnos.

Una se encuentra en el vestíbulo de la facultad (*véase Figura 1*), un lugar idóneo puesto que se encuentra entre la puerta delantera y la puerta trasera, prácticamente paso obligado para cualquier alumno y/o visitante.

Figura 1. Pantalla situada en el vestíbulo

La otra pantalla se encuentra en el otro punto más concurrido de la facultad, que es la cafetería (*véase Figura 2*).

Figura 2. Pantalla situada en la cafetería

Estas pantallas se utilizan para mantener informados a los alumnos, de manera complementaria a los correos electrónicos y a los carteles físicos, sobre diversas temáticas relacionadas con la facultad: consejos sobre seguridad informática, recordatorios como los plazos de convocatoria de becas, anuncios de eventos importantes y de interés como conferencias o concursos de programación, etc.

3.1. Sistema de gestión de pantallas inicial

La estructura de este sistema (*véase Figura 3*) está basada en dos partes fundamentales: un servidor que contiene el contenido a mostrar en las pantallas, y un ordenador ejecutando un reproductor el cual se comunica con el servidor y reproduce el contenido programado.

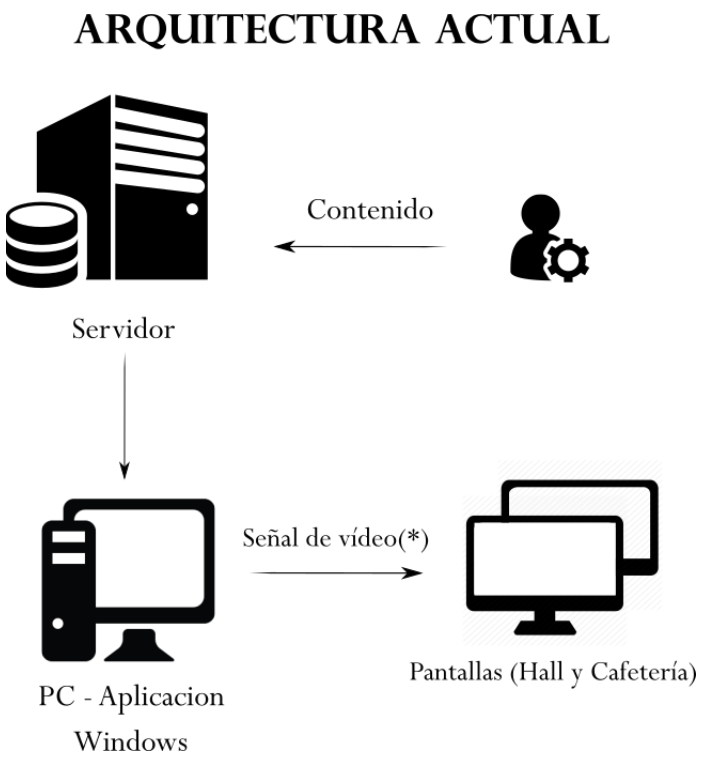

Figura 3. Arquitectura del sistema gestión inicial. (*) Las pantallas están conectadas directamente a la salida de vídeo del PC, por lo que puede aparecer el cursor o alertas del sistema operativo.

El servidor que gestiona el contenido se encuentra disponible las 24 horas del día y dispone de un CMS accesible desde el navegador a través del cual organiza este contenido. Solo los usuarios con privilegios pueden acceder a este CMS, ya sea desde la facultad o fuera de ella, y añadir, eliminar o modificar el contenido que se quiere mostrar en las pantallas.
En otro sitio de la facultad se encuentra el ordenador receptor del contenido, el cual como hemos dicho se limita a ejecutar el reproductor que se comunica con el servidor y muestra por pantalla los diferentes contenidos programados.

Con esta estructura, para modificar el contenido hay que acceder al CMS del servidor como usuario con privilegios, y una vez dentro podemos ver una lista con los diferentes contenidos, así como su fecha de inicio, fecha de finalización, prioridad, etc.
Para añadir un contenido debemos subir el fichero en cuestión al servidor, o la URL que queremos mostrar, y dar los valores requeridos a cada atributo. Pero, como decíamos en la introducción, este funcionamiento limita mucho a la hora de añadir un contenido dinámico, que no se limite a un fichero o una URL y que, aun siéndolo, no lo tengamos ni sepamos cuándo va a estar disponible. Así, ahora mismo resulta impensable mostrar por las pantallas de la facultad una retransmisión de un concurso de programación que tenga lugar en los laboratorios, ya que tendría que estar una persona realizando cambios constantes en algún fichero, subiéndolo al servidor de contenidos y rehaciendo estos pasos sin parar durante el tiempo que dure el concurso.

Para cambiar esta dinámica, y con el fin de encontrar una herramienta que se ajuste a las características deseadas para nuestro proyecto, realizamos un estudio de las tecnologías y programas actuales que se asemejen a lo que queremos, del cual recogemos lo más importante en los siguientes apartados.

3.2. Dashboard

La primera opción que contemplamos es el concepto de software de *Dashboard* [1](que en castellano se traduce como panel de instrumentos), el cual se utiliza para mostrar información en pantallas de manera dinámica, pero está destinado al sector empresarial.

Esta información se muestra principalmente en forma de gráficos, los cuales se forman a partir de gran cantidad de datos de carácter importante para una empresa, y sirven para valorar y mejorar la situación de una empresa y su estrategia. Debido a la gran cantidad de datos que manejan suelen ofrecer muchas funcionalidades relacionadas con su análisis.
En definitiva, el objeto de este tipo de software va destinado a la propia empresa que lo utiliza, ya que la información que muestra es aprovechada para valorar y reorganizar su actividad en función de sus objetivos.
Por este motivo, y tras analizar varias aplicaciones, decidimos descartar el uso de software de *Dashboard* ya que no se ajusta a lo que necesitamos; aun así, mencionamos aquí algunos ejemplos de este tipo de software:

- jSlate[3]:

  Es software de código libre, desarrollado principalmente en PHP y JavaScript. Se instala sobre LAMP.

  Su funcionamiento consiste en la organización de paneles, los cuales constan de widgets, para mostrar datos de manera gráfica.

  Como puntos a favor, ofrece plantillas de widgets, y estos widgets son desarrollados con tecnologías web como JavaScript, lo cual facilita la creación y personalización de los mismos.

  El principal inconveniente para nosotros es que estos widgets están enfocados exclusivamente a la creación de múltiples tipos de gráficos para la representación de datos.

- Freeboard[4]:

  Es software de código libre (licencia MIT), basado en *JavaScript*. Se puede instalar tanto en Windows como en Mac o sistemas operativos basados en Linux.

  Una herramienta sencilla de usar, que te permite gestionar contenido en paneles (*véase Figura 4*) de un ancho fijo y que van formando un *"grid"* con ellos. Soporta contenido de distintos tipos, como imágenes, URLs, etc. También permite obtener contenido de un sitio remoto mediante JSON o similar.

[3] jSlate: https://github.com/rasmusbergpalm/jslate

[4] Freeboard: https://github.com/Freeboard/freeboard

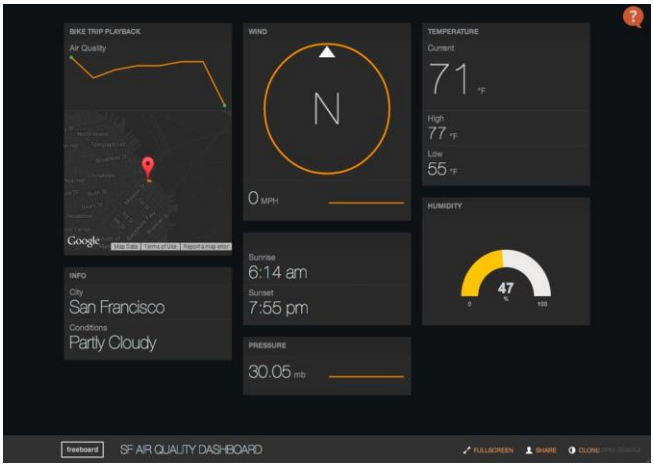

Figura 4. Freeboard: Interfaz principal

Sus principales ventajas son su sencillez y la facilidad con la que se consigue la persistencia de sus datos, que consiste en una función de guardar el tablero (que incluye el contenido), para su posterior carga manual.

Los principales inconvenientes de esta herramienta son dos: no se puede redimensionar los paneles al gusto del usuario, y tampoco permite el control del contenido a mostrar por tiempo, aunque sí permite refrescar las fuentes de datos cada cierto periodo. Cuenta con una aplicación de prueba[5].

- PHP Dashboard[6]:

  Es software privativo, ofrece una versión de prueba de 30 días. Está desarrollado en PHP, y se ejecuta sobre Apache. Puede ser instalado en sistemas operativos Windows o basados en Linux.

  Consiste en una aplicación web, y su funcionamiento se basa en una herramienta de arrastrar y soltar desarrollada con JavaScript que facilita la experiencia del usuario. Puede generar multitud de tipos de gráficos (estáticos o dinámicos), y admite una gran cantidad de tipos de bases de datos diferentes como fuente de sus datos.

  Sus principales ventajas son su simplicidad y facilidad de uso para el usuario inexperto, y la amplia conectividad con tipos de bases de datos para el usuario técnico.

  La principal desventaja es que no es de uso gratuito.

---

[5] Pruebas Freeboard: http://freeboard.github.io/freeboard

[6] PHP Dashboard: https://dashboardbuilder.net/php-dashboard

3.3. Digital Signage

El *Digital Signage* [2] (señalización digital dinámica o cartelería digital) es un concepto moderno de publicidad [3] basado en la emisión de contenido digital, de manera programada, en pantallas. Este contenido a emitir puede ser de muchos tipos: texto, imágenes, vídeos (incluso en streaming), calendarios, información sobre el clima, tweets, etc.

A diferencia del software de Dashboard, su utilidad no va destinada a la propia empresa que lo utiliza, sino que va destinado a un público externo a la propia empresa.

Actualmente podemos encontrar ejemplos de uso de la cartelería digital en multitud de lugares públicos, como los que a continuación enumeramos:

- Museos en los que sus pantallas muestran a sus visitantes información y disponibilidad de las diferentes salas.
- Restaurantes que muestran a sus clientes sus últimas promociones y/o nuevos productos por pantalla.
- Tiendas que añaden pantallas en los escaparates donde mostrar sus diseños para la nueva temporada, con el fin de atraer a clientes potenciales entre quienes pasan por delante.

Teniendo en cuenta los múltiples usos de la cartelería digital, y poniendo estos posibles usos en común con los objetivos que nos marcamos al principio, llegamos a la conclusión de que este tipo de software es el que necesitamos para nuestro proyecto. Además, existen bastantes aplicaciones que cuentan con versiones estables (pese a ser un concepto reciente, se está avanzando rápido en su expansión, y sobre todo nos permite trabajar con contenido dinámico y en tiempo real). El primer impedimento que encontramos es que la mayoría son de tipo privativo. Pese a ello, citamos a continuación algunos:

- Mural Digital[7] de Vixonic:

  Basado en la tecnología cliente-servidor utilizada para mostrar imágenes, vídeo y texto.

  Dispone de dos tipos de módulos según la manera en la que su contenido se actualiza: manuales, los cuales requieren de una asistencia permanente, y

[7] Mural Digital: https://www.vixonic.com/mural-digital/

automáticos, que están enlazadas a una fuente de datos que se actualiza automáticamente.

Figura 5. Mural digital de Vixonic

Basa su diseño en el uso de diferentes apps prefijadas que se reparten la pantalla para mostrar su contenido en diferentes regiones (*véase Figura 5*).

- Smush Digital[8]:

  Esta herramienta ofrece un servidor que incluye un gestor de contenido alojado en la nube, por ello su oferta se basa en ser un *Software As a Service*. Este contenido se gestiona desde un programa y se reproduce desde otro, los cuales tienen versiones compatibles con Windows, Mac y Android.

  La solución completa de digital signage que nos ofrece divide su estructura en tres partes:

  - Smush Digital Studio: software que controla y emite las órdenes que se le dan al servidor de contenidos.
  - Servidor de contenidos: donde se almacenan los contenidos y las órdenes sobre los mismos recibidas del Smush Digital Studio.
  - Smush Digital Player: encargado de recibir y emitir los contenidos.

Como preferimos utilizar una herramienta de software libre que se ajuste a nuestras necesidades, rápidamente descartamos entre otros los anteriores, y nuestra búsqueda se reduce a las aplicaciones con licencia de software libre. Aquí citamos algunas:

[8] Smush Digital: https://www.smushdigital.com/signage-software

- Concerto[9]:

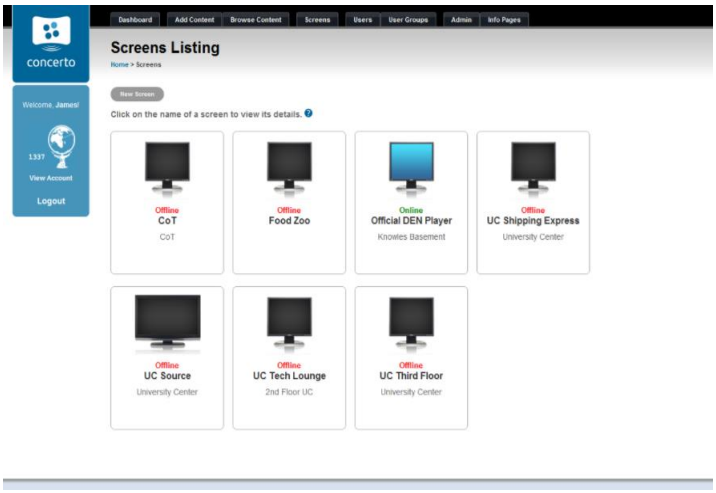

Figura 6. Concerto: GUI Administración Pantallas

Está implementado con código Ruby (principalmente) y HTML. Ofrece dos tipos de instalaciones: una para sistemas operativos basados en Linux (una en especial para Debian), y otra en formato imagen de máquina virtual.

Utiliza servidores a los que usuarios mandan contenido de un cierto tipo (gráfico, textual u otro) y los usuarios moderadores se encargan de aceptar y mostrar ese contenido en las pantallas conectadas a la interfaz de Concerto (*véase Figura 6*).

- DisplayMonkey[10]:

  Bajo licencia de tipo MIT, está desarrollado principalmente en JavaScript y C#. Está preparado para instalarse en productos de Microsoft: su sistema operativo Windows y SQL Server.

  Su funcionamiento consiste en una aplicación basada en navegador, y divide su estructura en dos aplicaciones: una de administración de contenidos (DMM) que se encarga de qué mostrar en las pantallas y otra de presentación (DMP) de contenidos que permite diseñar cómo mostrar el contenido (*véase Figura 7*).

[9] Concerto: https://github.com/concerto/concerto

[10] DisplayMonkey: https://github.com/fuel9/DisplayMonkey

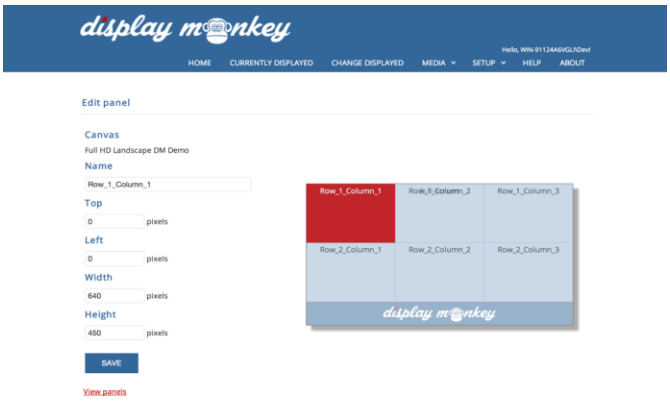

Figura 7. DisplayMonkey: Edición de paneles

- Xibo[11]:

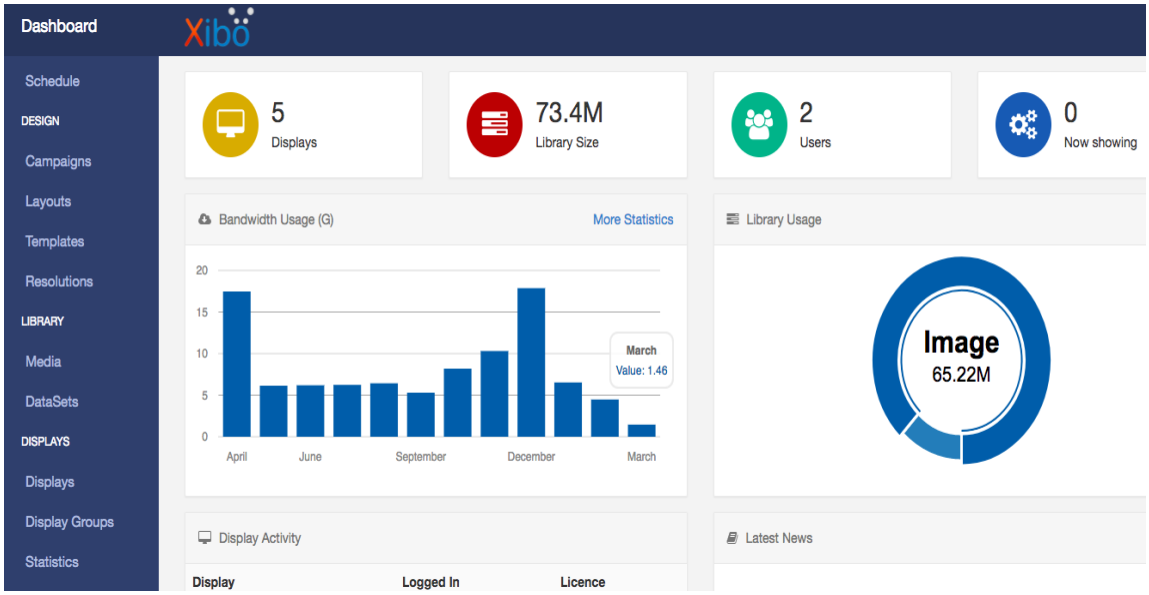

Figura 8. Xibo: Pantalla Principal

Su licencia es AGPLv3.

Utiliza diferentes widgets para los distintos tipos de contenido con los que puede trabajar, además de ofrecer el uso de plantillas. Se pueden crear widgets nuevos totalmente personalizables basados en PHP y HTML.

[11] Xibo Github: https://github.com/xibosignage

Divide su funcionamiento en dos partes:

- Content Management System (CMS) (*véase Figura 8)* desarrollado principalmente en PHP, HTML y JavaScript. Versiones disponibles:
  - En la nube, como *Software as a Service*.
  - Auto-hosted: sobre un servidor propio, ya sea basado en Docker o con servidor PHP/MySQL.
- Reproductor: versiones disponibles para Windows (en C#), Android y WebOS.

Tras probar algunas de estas y otras aplicaciones Digital Signage de software libre, decidimos empezar a trabajar con, Xibo, de la empresa Xibo Signage LTD. Es una empresa que cuenta con años de experiencia, y su producto está siendo muy utilizado y contrastado.
En su página web[12] podemos ver que es una solución utilizada en bastantes empresas, además de poder leer artículos de opinión de la gente que lo utiliza. Sobre esto, cabe destacar, por la cercanía del caso de uso, la utilización de Xibo en la Università degli di Urbino Carlo Bo de Urbino (Italia) [4], y en la Longhill High School de Rottingdean (Inglaterra) [5].

Para este proyecto hemos utilizado la versión estable 1.8.12 del CMS de Xibo, así como la misma versión de su reproductor para Windows. En cuanto a las versiones disponibles, cabe destacar que cuando nosotros empezábamos este proyecto desde Xibo estaban ultimando el desarrollo de la versión 2.0 de su CMS, la cual no hemos probado puesto que supondría un cambio que podría comprometer la estabilidad de nuestro sistema, así como una versión en fase alpha para Linux de su reproductor que sí pudimos probar, la cual contaba todavía con muchos fallos característicos por su estado de desarrollo.

A continuación, pasamos a describir la herramienta Xibo en detalle, tanto su estructura como su funcionamiento, así como el uso que podemos darle y las partes que podemos desarrollar con ella.

[12] Xibo Webpage: https://xibo.org.uk/

Capítulo 4. Xibo

El conjunto de software de Xibo ofrece una herramienta completa de Digital Signage: mediante tecnología cliente-servidor cuenta por el lado del servidor con un CMS que gestiona el contenido a mostrar en las pantallas, y por el lado del cliente un reproductor que es capaz de conectarse con el CMS y reproducir este contenido.

4.1 El CMS

El CMS, que permite controlar el contenido que se debe mostrar, se instala en un web server Apache, sobre Docker o un servidor propio PHP/MySQL.
Xibo también ofrece la opción de pago de tener el CMS montado en la nube en sus propios servidores, con un mes de prueba gratuito.

Una vez logeado, desde el Panel de Control podemos observar todos los recursos que Xibo ofrece, así como un resumen estadístico de su uso: pantallas conectadas, usuarios, ancho de banda empleado, información sobre la biblioteca multimedia que incluye y últimas noticias sobre Xibo.
A continuación, describiremos las funciones más importantes que ofrece.

4.1.1 Diseño de contenidos

En la sección de Diseño, el CMS de Xibo se basa en la existencia de Layouts o diapositivas, entidad sobre la cual se sustenta su funcionamiento.
Una diapositiva representa a la pantalla, y como tal se llena del contenido que pretendemos que se muestre. Para ello, cada diapositiva se compone estructuralmente de una o varias regiones donde colocar el contenido, y funcionalmente de widgets, que son los elementos que lo llenan de contenido (*véase Figura 9*).
Estos widgets pueden ser de diversos tipos, según el tipo de contenido a reproducir: imágenes, vídeos, url's, contenido embebido, calendarios de Google, tweets, PDF's, PowerPoint, etc. Cada tipo de widget diferente se corresponde con una entidad existente por debajo, llamada Módulo.

Cada módulo consiste en un directorio situado en una ruta concreta relativa a la raíz de la instalación del CMS, en la que debe haber diferentes ficheros que conforman la estructura y el diseño de los distintos widgets y de sus formularios de configuración, adición y edición.

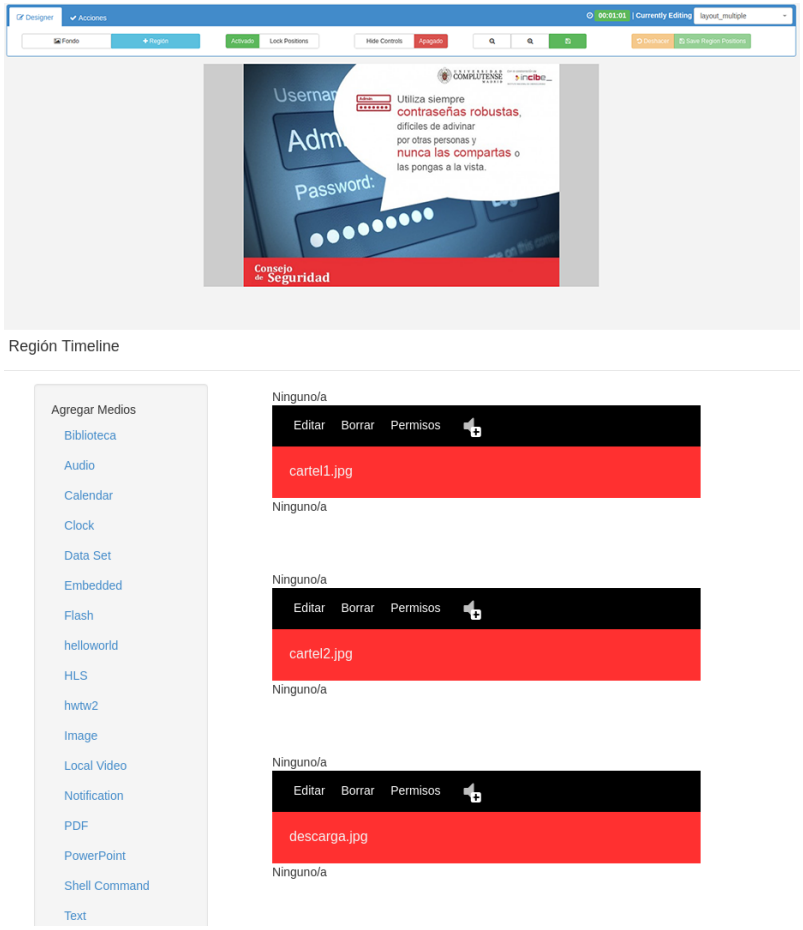

Figura 9. Xibo: Panel edición de regiones y panel edición de widgets

Además, existe una entidad llamada Campaign o campaña, que simplemente es un conjunto de diapositivas. Al crear una campaña se seleccionan las diapositivas que la componen y el orden en el que se sucederán cuando la campaña sea programada.

El CMS de Xibo también permite la creación de Plantillas de diapositivas. Esta opción puede ser útil para la creación posterior de diapositivas que guarden una estructura similar, como podría ser mostrar tweets o fecha y hora a un lado, y un contenido central diferente para cada una.

Para gestionar estos contenidos, el CMS de Xibo cuenta con una Biblioteca donde guardar los ficheros necesarios, como vídeos o imágenes.

4.1.2 Programación de contenidos

En cuanto a la Programación en las pantallas, la entidad imprescindible en Xibo es el Event o evento. Cuando se crean eventos (*véanse Figuras 10 y11*), a cada uno de se le agrega una diapositiva o una campaña con el contenido a reproducir, sus correspondientes fecha y hora de inicio y fin de reproducción, y se le asigna una pantalla o grupo de pantallas (ver en el siguiente apartado). Además de estos campos, es necesario también asignarle una prioridad y un orden, que serán decisivos a la hora de elegir qué y cuándo reproducir cada evento cuando existen más de uno programados. La elección del evento a reproducir sigue el siguiente esquema:

- Se reproduce siempre el evento con mayor prioridad
- Si hay varios eventos con la misma prioridad en el mismo periodo temporal se reproducen en el orden designado por el parámetro "orden". La duración de cada evento antes de pasar al siguiente viene dada por el tiempo de reproducción del contenido del evento.
- Si hay un evento reproduciendo y entra uno nuevo con prioridad mayor, el entrante expropiará al de prioridad menor entrando en reproducción.

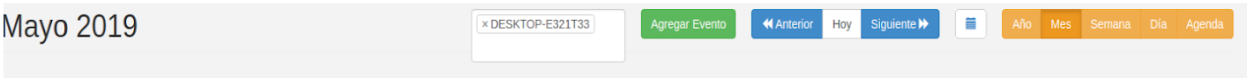

Figura 10. Xibo: Panel de edición de eventos

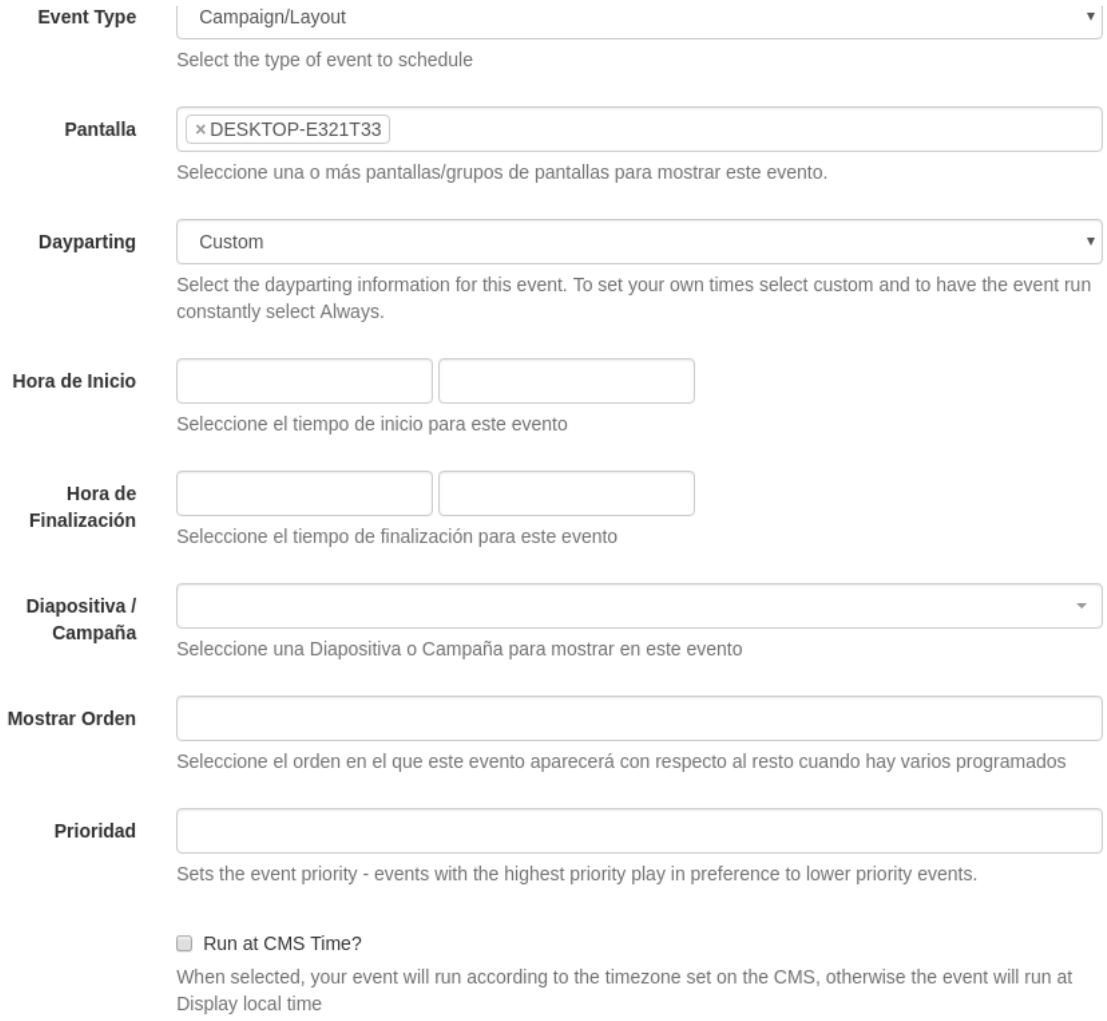

Figura 11. Xibo: Programación nuevo evento

4.1.3 Conexión con las pantallas

En la sección de Pantallas se gestionan las pantallas conectadas al CMS, siempre ligadas a un reproductor (software del cual hablaremos más adelante). Aquí la función más importante es la de autorizar una pantalla, sin la cual no será posible la reproducción del contenido.

Además, Xibo ofrece la posibilidad de crear grupos de pantallas, para los cuales asignar eventos en común. Esto puede venir bien cuando se tiene diferentes reproductores conectados al CMS que deben tener la misma programación.

La opción de Configurar pantallas permite gestionar el perfil de un reproductor, existiendo un perfil prefijado por cada sistema operativo diferente (Android, WebOS, Linux, Tizen y

el que nos ocupa, Windows), y permitiendo también editarlos y añadir nuevos perfiles personalizados. En este perfil se incluye el Intervalo de recolección, que es el tiempo que tarda el reproductor en conectar con el CMS para traerse el nuevo contenido y reproducirlo. Se puede usar esta opción, o activar otra de las características que ofrece Xibo: el XMR (Xibo Message Relay), el cual deja en un segundo plano el intervalo de recolección y directamente se encarga de mandar el contenido del CMS al reproductor cada vez que se programe un nuevo evento.

4.1.4 Administración

En la parte de administración hay que destacar la existencia de la opción Módulos. En esta parte se muestran todos los módulos instalados en el CMS, pudiendo habilitarlos o deshabilitarlos, y permite instalar módulos ya configurados y preparados para su instalación, o nuevos módulos desarrollados principalmente en PHP y HTML (también pueden incluir JavaScript para la parte de diseño) que cumplan los requisitos necesarios para que el CMS los detecte como módulos instalables.

4.2 El reproductor

El reproductor, que es el software que debe instalarse para reproducir el contenido, se trata de un programa que se comunica con el CMS y muestra el contenido programado por el mismo. Este programa debe estar ejecutándose en un ordenador a modo de receptor, y se limita a reproducir en su pantalla (por defecto a pantalla completa) los eventos que desde el CMS se hayan programado para este reproductor en cada momento.
En nuestro caso, el ordenador que ejecute este programa deberá estar conectado a las pantallas para reproducir el mismo contenido.

Actualmente existen tres versiones estables: una para Windows, otra para WebOS y otra para Android. Por esto, y aunque instalamos y probamos la versión en desarrollo de Linux, su inestabilidad y falta de funcionalidad sobre ciertos widgets nos hace decantarnos por la versión estable de Windows para la reproducción del contenido programado de nuestro CMS.

El funcionamiento del reproductor es muy sencillo: se rellenan los campos necesarios de la pestaña "Connect", y se lanza una petición al CMS al pulsar el botón "Save". Una vez hecho esto, si todos los campos eran correctos, en el CMS se quedará registrado nuestro reproductor dentro de la sección Pantallas, y desde ahí habrá que habilitarla para que próximamente al pulsar el botón "Launch Client" de nuestro reproductor se reproduzca en modo pantalla completa (por defecto) el contenido programado desde el CMS (o la diapositiva por defecto en caso de que no haya ningún contenido programado para esta pantalla).

4.3 La API del CMS

El CMS de Xibo cuenta con una API REST de servicios bastante completa, la cual utilizamos para desarrollar nuestro programa que hace de controlador de contenidos y mediante llamadas a esta API gestiona el contenido del CMS. Esta API sigue la especificación OpenAPI y presenta su documentación [6] haciendo uso de Swagger-UI.

Partiendo de la ruta del CMS, la API se ubica en "`/api`". Para acceder a ella hay que cumplir con el protocolo de autorización oAuth. Para ello, permite dos tipos de acceso autenticado: mediante código de acceso ("`access_code`") o el que usamos nosotros, con credenciales de cliente ("`client_credentials`"), que consiste en registrar en el CMS la aplicación que hace uso de la API y antes de nada realizar una petición con estas credenciales de acceso al método de autorización, que devolverá un código de acceso ("`acces_token`") con el que podremos realizar peticiones a todos los métodos disponibles.
Una vez autorizado la API ofrece métodos para la obtención, adición, edición y eliminación de la mayor parte de las entidades con las que trabaja el CMS. Los principales métodos que nos han servido para la conexión de nuestro controlador con el CMS son los que trabajan con diapositivas, widgets, campañas, eventos y librería.

A continuación, recogeremos los pasos a seguir para tener una instalación de los componentes necesarios de Xibo para el funcionamiento de nuestro sistema.

4.4 Guía de instalación de Xibo

Como hemos explicado previamente, la herramienta digital signage de Xibo se compone de dos partes fundamentales: el CMS que gestiona el contenido, y el reproductor que lo reproduce.

Aunque en nuestro proyecto no contemplamos la opción de utilizar el CMS de Xibo en sus servidores, cabe destacar que esta es una solución a tener muy en cuenta. En nuestro caso nos ha servido para hacer uso del soporte técnico de Xibo sobre un reproductor de una manera más directa y específica que el contacto en su foro. Para conseguirlo, hay que seguir las instrucciones que dan en su página[13].

En los siguientes puntos describiremos de forma breve y concreta los pasos necesarios para la instalación de los diferentes softwares de Xibo que hemos utilizado: el CMS sobre servidor propio, y el reproductor de Windows. Además, describiremos los pasos a seguir para lograr una comunicación rápida entre ambos.

4.4.1 Instalación del CMS

Los siguientes pasos indican el proceso para la instalación del CMS 1.8.12 de Xibo empaquetado con Docker sobre un sistema operativo basado en Linux, en nuestro caso Ubuntu 18.0.4.

Lo primero, y como requisito previo antes de proceder a la instalación, consiste en tener instalada la parte que necesitamos de Docker para poder montar el CMS: una instalación válida de Docker y Docker-Compose. (Versión mínima 1.23.0)
Pasos para instalar el CMS desde terminal:

1. Crear el directorio raíz de la instalación de Xibo-CMS.
   Recomiendan hacerlo en "`/opt/Xibo`".

```
~$ sudo mkdir Xibo-Docker-CMS
~$ cd Xibo-Docker-CMS
```

[13] Xibo en la nube: https://xibo.org.uk/docs/setup/xibo-in-the-cloud

2. Descargar el fichero de instalación del github de Xibo (en la ruta se especifica la versión).
   `~/Xibo-Docker-CMS$ sudo wget https://github.com/xibosignage/xibo-cms/releases/download/1.8.12/xibo-docker.tar.gz`
3. Extraer el contenido del fichero comprimido.
   `~/Xibo-Docker-CMS$ sudo tar -xf xibo-docker.tar.gz`
   `~/Xibo-Docker-CMS$ cd xibo-docker-1.8.12/`
4. Modificar el fichero principal de configuración "`config.env.template`".
   El único parámetro obligatorio es establecer la contraseña del usuario de la base de datos. Podemos configurar otros parámetros opcionales como el servidor SMTP, opciones del servidor web como el nombre de dominio o la ruta del CMS (alias), además de otras opciones.
   Cuando hayamos acabado, debemos guardar el fichero como "`config.env`" (eliminando "template" del final).
5. Se pueden configurar otros aspectos del CMS como puertos personalizados, opciones de MySQL y HTTPS/SSL. Existen ficheros "`*.template`" para todas estas opciones. Nosotros mantenemos la configuración predeterminada que nos ofrece Xibo.
6. Instalar los contenedores del CMS. Ejecutar:
   `~/Xibo-Docker-CMS/xibo-docker-1.8.11$ sudo docker-compose up -d`
7. Tras el tiempo que necesite para la instalación y montaje del contendor, ya podemos acceder al CMS mediante "`localhost[:PORT]/`" (o "`localhost[:PORT]/<alias>`" en caso de haber modificado el alias en el paso 4) en un navegador.

Parar, arrancar y reiniciar el servicio de Xibo:

```
~/Xibo-Docker-CMS/xibo-docker-1.8.11$ sudo docker-compose stop
~/Xibo-Docker-CMS/xibo-docker-1.8.11$ sudo docker-compose start
~/Xibo-Docker-CMS/xibo-docker-1.8.11$ sudo docker-compose restart
```

Para desinstalar el CMS de Xibo, junto a todos sus componentes: Web server, MySql...

```
~/Xibo-Docker-CMS/xibo-docker-1.8.11$ sudo docker-compose down
```

4.4.2 Instalación del reproductor

Los siguientes pasos indican el proceso para la instalación del reproductor de Xibo sobre el sistema operativo Windows en su versión 10.

1. Descargar el instalador[14], ejecutarlo y seguir los típicos pasos de instalación.
2. Una vez instalado abrir Xibo Player Options y configurar:
   - ➢ CMS Address: dirección (local o externa) de tu CMS.
   - ➢ Key: se localiza en el CMS: Administración -> Preferencias
   - ➢ Local Library: ruta donde guardar una copia del contenido en local, permitiendo así que el reproductor funcione aunque pierda la conexión con el CMS.
3. Pulsar el botón "Save".

Tras esto, se mostrará un mensaje de confirmación a la espera de que el reproductor sea autorizado desde el CMS. Por tanto, solo nos quedaría:

4. Desde el CMS: Pantallas -> Opciones de la pantalla que acabamos de añadir -> Authorise

4.4.3 Cómo habilitar XMR

En esta versión del CMS con Docker XMR, que sirve para que el CMS avise al reproductor sobre cambios en el contenido que debe reproducir, está preinstalado y parcialmente configurado. Para terminar de configurarlo basta con acceder al CMS y en "Configurar pantallas" editar el perfil Windows y modificar la opción "Dirección Pública XMR" añdiendo "`tcp://ipDelCMS:9505`".

[14] Última versión del reproductor de Xibo: https://xibo.org.uk/api/downloads/windows

Capítulo 5. Controlador de gestión diaria

En este capítulo explicamos nuestra herramienta desarrollada para la gestión diaria de los contenidos de las pantallas, incluyendo la motivación para desarrollarla y los objetivos marcados, y el apartado técnico de la misma, así como su uso y guía de instalación.

5.1. Motivación y objetivos

La idea de este controlador surgió para poder agrupar varias acciones del CMS rápidamente. Esto permitiría a la/s persona/s encargada/s de programar el contenido hacerlo sin estar accediendo de forma directa al propio CMS. De esta forma se elimina la necesidad de aprender a utilizar el CMS desde cero, para programar contenido de una manera más rápida y simple.

Esta aplicación permite hacer uso, en pocos pasos, de las funcionalidades básicas de Xibo, consideradas como necesarias para el día a día en la facultad.

Un ejemplo práctico de uso podría llevarse a cabo un día con condiciones meteorológicas adversas. El responsable de administrar las pantallas accedería a la aplicación y desde ahí podría programar una diapositiva en la que se informa de la cancelación de las clases. También sería capaz de ver el resto de los eventos programados para ese periodo de tiempo, por lo que podría elevar la prioridad del aviso para que este nuevo evento expropie la programación actual y sea el único que se visualice en las pantallas.

5.2. Arquitectura

Esta aplicación se ha implementado con el *framework* Express.js[15]. Consta de un servicio web en el que se pueden observar diferentes módulos sobre una arquitectura Modelo-Vista-Controlador (MVC) (*véase Figura 12*).

[15] Express.js: https://expressjs.com/es/

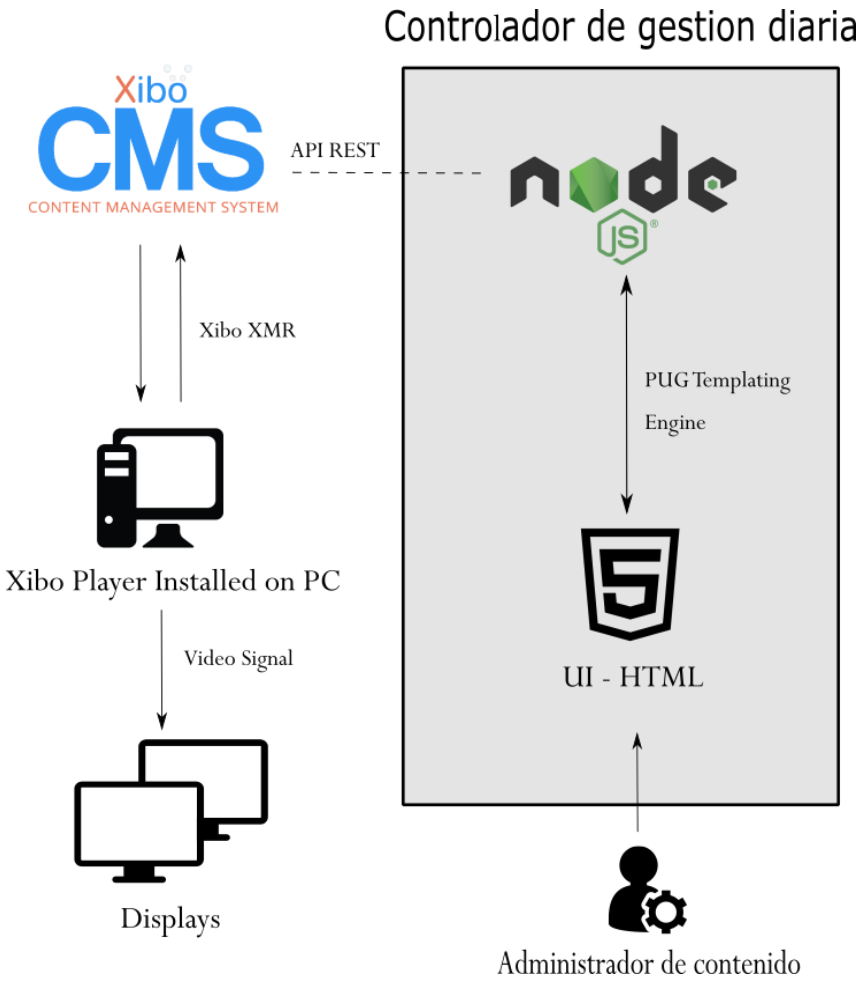

Figura 12. Arquitectura del Controlador de Gestión Diaria

El código se estructura de la siguiente manera:

- Servicios Web

  En este módulo se realizan las peticiones a la API REST de Xibo. Es el "modelo" en nuestra arquitectura.

  Xibo dispone de un servidor de autenticación OAUTH2. Por lo tanto, para realizar cualquier petición a través de la API necesitamos obtener un token de acceso. Este token puede obtenerse a través de dos métodos diferentes: mediante credenciales de acceso y mediante un código de autorización. Nuestra aplicación accede mediante credenciales de acceso. Estas credenciales se obtienen del CMS y constan de un id de cliente y un secreto (para cada aplicación cliente). De esta forma se puede controlar el nivel de acceso al CMS desde clientes externos.

Una vez hemos obtenido el token de acceso, debemos incluirlo en la cabecera de cada petición que hagamos al CMS, teniendo en cuenta que el token expira en una hora.

- Controladores

  En esta capa se incluye la lógica con la que se tratan los datos (procedentes de la capa de Servicios o de la entrada del usuario). Existen controladores para operar con cada entidad importante de Xibo: Layout, Campaign y Event (entidades descritas en el capítulo 4 de este documento).

- Vistas:

  Conforman la interfaz de usuario. Están implementadas en HTML5.

5.3. Casos de uso

El usuario autenticado de nuestra aplicación es el encargado de administrar las pantallas. La autenticación se lleva a cabo mediante un formulario básico, de nombre de usuario y contraseña, en la aplicación. Las funciones que puede realizar el usuario con nuestra aplicación se pueden categorizar por la entidad principal con la que tratan, como explicamos en los siguientes subapartados, y por ello el menú principal se divide en tres: Layout Manager, Campaign Manager y Event Manager.

5.3.1 Gestión de diapositivas

A la gestión de diapositivas está dedicada la opción Layout Manager (*véase Figura 13*). La interfaz se divide en tres partes, según su funcionalidad: la zona superior izquierda permite crear nueva una diapositiva a partir de un nombre introducido, la zona inferior izquierda permite eliminar una diapositiva existente a partir de su ID y, por último, la zona de la derecha muestra una lista actualizada de las diapositivas existentes en nuestro CMS y sus respectivos IDs.
Además, si se selecciona una de las diapositivas de la lista y se pincha en el botón "`Design`", es posible añadir o eliminar widgets en la diapositiva seleccionada. Para ello hay una redirección a una vista diferente (*véase Figura 14*) en la que se observan dos zonas diferenciadas: en la parte superior se muestra una lista de los widgets que contiene

la diapositiva, mientras que la parte inferior ofrece un desplegable con diferentes tipos de widgets para insertar.
Al intentar insertar un widget, se muestra una vista diferente en la que simplemente hay que rellenar los campos necesarios para añadir al widget.

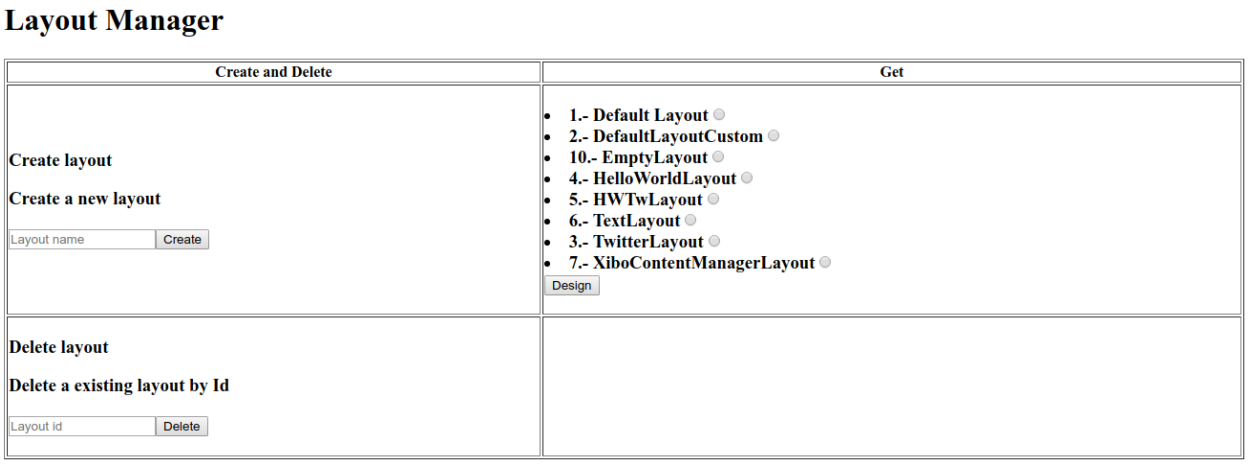

Figura 13. Interfaz gestor de diapositivas

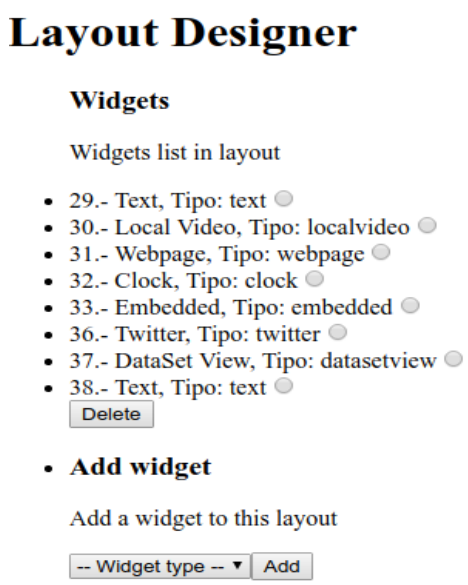

Figura 14. Interfaz diseñador de diapositivas

5.3.2 Gestión de campañas

La opción Campaign Manager (*véase Figura 15*) se dedica a la gestión de campañas, y presenta un diseño y una funcionalidad muy semejante a la del Layout Manager explicado en el apartado anterior.
Como antes, la parte superior izquierda permite insertar una nueva campaña a partir del nombre que se le quiera dar, y la parte inferior izquierda permite eliminar una campaña existente a partir de su ID. La parte derecha muestra una lista actualizada de campañas existentes en el CMS.

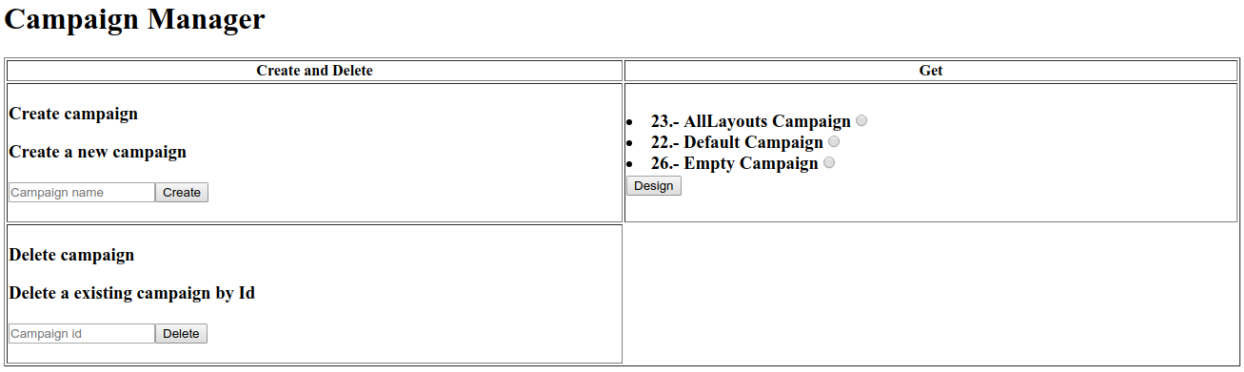

Figura 15. Interfaz gestor de campañas

5.3.3 Gestión de eventos

La gestión de eventos se lleva a cabo en el Event Manager (*véase Figura 16*), cuya interfaz se divide en dos partes: a la izquierda se encuentra el formulario de creación de evento, y a la derecha una lista de los eventos programados actualmente e información importante sobre cada uno de ellos: su ID de evento, su prioridad y orden (es trascendental a la hora de programar nuevos eventos en la misma franja de tiempo) y el ID del contenido que tienen programado (diapositiva o campaña).
Además, debajo de esta lista de eventos es posible eliminar uno existente (aunque no esté programado actualmente), a partir de su ID.
Para insertar un evento basta con elegir una diapositiva o una campaña de entre las que hay en la lista, seleccionar o introducir las fechas y horas en que se desea programar el inicio y el fin del evento, y asignarle manualmente una prioridad y un orden numéricos.

La pantalla que reproducirá este nuevo evento viene definida por una constante el fichero de configuración del sistema (*véanse Figuras 17, 18, 26 y 27*), de la cual hablaremos en el siguiente punto.

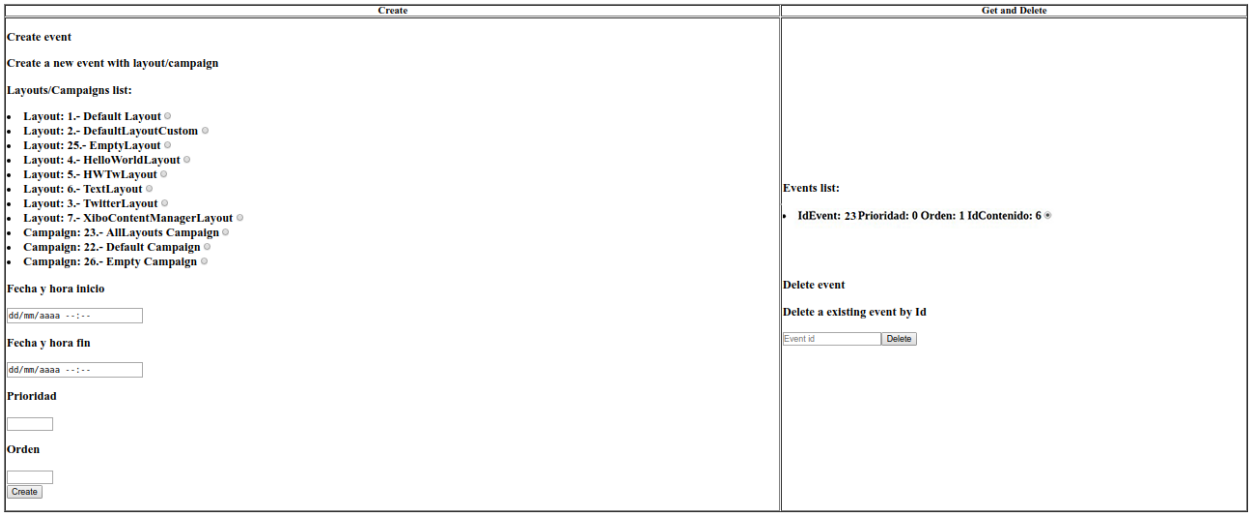

Figura 16. Interfaz gestor de eventos

5.4. Despliegue e instalación:

5.4.1 Configuración necesaria:

Prerrequisitos:

- Node.js (v10.15.3-LTS at 20/05/19): https://nodejs.org/es/
- NPM (funcionando en v6.4.1): https://www.npmjs.com/

En este apartado recogemos la configuración, tanto de nuestra aplicación como CMS de Xibo, para garantizar el correcto funcionamiento del sistema.

Es necesario realizar los siguientes pasos en el orden indicado:

1. Descargar el código fuente desde aquí: https://github.com/TfgDigitalSignage/Xibo-Content-Manager (Rama master)

2. Desde la raíz, ejecutamos el comando `npm i` para descargar todas las dependencias

3. Registro de la aplicación en el CMS:

   En el CMS de Xibo hay que acceder a la opción Aplicaciones que se encuentra en la zona de Administración. Se crea una aplicación y, posteriormente, se edita para obtener los parámetros de autorización "Client ID" y "Client Secret". Además, hay que marcar la opción "Client Credentials" y desmarcar el resto, y en la pestaña de Permisos hay que marcar la opción "All access".

   Este proceso dará de alta un usuario en la base de datos de Xibo con el cual podremos autenticarnos para realizar peticiones a la API.

4. Otorgar a nuestra aplicación las credenciales de acceso a la API:

   En el directorio raíz del proyecto existe un fichero llamado "`.env`". Ese es el fichero de configuración de la aplicación. Se carga al lanzarse el servidor y contiene toda la configuración relevante para que no falle ningún subsistema. Para este paso, configuramos las *claves* de la sección de Xibo (*véase Figura 17*).

```
#Xibo CMS Options
#-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------
## SET APPLICATION ID/SECRET PREVIOUSLY CREATED ON XIBO CMS.
## THIS APPLICATION NEED 'CLIENT CREDENTIALS' AUTH TYPE
## API ENDOPOINT MUST END ON '/api/'
## DISPLAY/DISPLAY GROUP ID ON XIBO CMS. After Configure Display or display group
## PRIORITY OF SCHEDULE TO BE CREATED ON COMPETITION. SELECT HIGHER NUMBER THAN EVENTS SCHEDULED AT THE
MOMENT COMPETITION STARTS

XIBO_CLIENT_ID=
XIBO_CLIENT_SECRET=
XIBO_API_URL=
XIBO_DISPLAY_ID=
XIBO_COMPETITION_PRIORITY=
```

Figura 17. Fichero de configuración: Sección parámetros de Xibo

   En `XIBO_CLIENT_ID` y `XIBO_CLIENT_SECRET` debemos introducir las cadenas alfanuméricas que ha generado Xibo al dar de alta la aplicación en el paso 1. En `XIBO_API_URL` debemos introducir la dirección IP completa de la máquina que en la que se encuentra instalado el CMS de Xibo (si está en la misma en la que se va a alojar el Node.js puede quedarse como "`http://localhost/api/`"). Es de suma importancia que dicha dirección acabe en "`/api/`". En la clave `XIBO_DISPLAY_ID` debemos introducir el grupo de pantallas previamente instaladas como se indica en los apartados 4.4.1 y 4.4.2 sobre la instalación de Xibo.

5. Configurar las claves de acceso a la aplicación:

```
#Server Configuration
#-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------
## USERNAME/PASSWORD OF ADMIN CREDENTIALS TO LOGIN IN THIS APPLICATIONS
ACCESS_USERNAME=
ACCESS_PASSWORD=
SESSION_SECRET=
```

Figura 18. Fichero de configuración: Sección parámetros del Controlador Gestión Diaria

   En esta sección es necesario es añadir un `ACCESS_USERNAME` y `ACCESS_PASSWORD`, además de un `SESSION_SECRET` cualquiera que sirve para la autenticación (*véase Figura 18*). Estos valores van a ser el usuario y la contraseña de acceso a la aplicación.

6. En este punto ya estamos listos para lanzar la aplicación. Ejecutamos el comando `npm start` sobre la raíz del proyecto.

Capítulo 6. Controlador de concursos

Como ya se ha introducido en el capítulo 1 de este documento, un uso práctico de nuestra aplicación trata de la gestión dinámica en tiempo real de contenido relacionado con los concursos de programación que se llevan a cabo periódicamente en la FDI. Un concurso de programación es una competición en la que participan varios equipos tratando de resolver un número concreto de problemas en el menor tiempo posible. Los equipos envían las soluciones de los problemas a un sistema de gestión de concursos que valida el código y otorga una puntuación, generando un "ranking" en el que, al finalizar el tiempo estipulado, el equipo con más puntos gana el concurso. Un ejemplo claro es el ICPC[16] o *International Collegiate Programming Contest* por sus siglas en inglés. Es una competición de programación entre Universidades de todo el mundo organizada por IBM.

6.1. Objetivo y motivación

Nuestra aportación en este aspecto ha sido implementar una aplicación capaz de controlar el flujo de un concurso gestionando la información más útil a mostrar, mediante Xibo, en cada momento.

Además de la información relativa al concurso, la aplicación obtendrá un *streaming* de video (véase sección 6.2.2 de este mismo capítulo), procedente de webcams instaladas en los ordenadores utilizados por los equipos durante la competición, con la finalidad de mostrar su reacción en el momento en el que el sistema de gestión de concursos otorga un veredicto a uno de sus envíos.

Nuestro objetivo es poner a prueba nuestro sistema durante el desarrollo de un concurso de ProgramaMe[17] que tendrá lugar en la FDI durante el mes de junio. El sistema de gestión de concursos a utilizar en este caso se llama DOMJudge[18] y es ya conocido por

[16] ICPC, *International Collegiate Programming Contest*: https://icpc.baylor.edu/
[17] ProgramaMe: http://www.programa-me.com/2019/nac/
[18] DOMjudge, *Universidad de Utrecht*: https://www.domjudge.org/

alumnos y profesores de la facultad. Está compuesto por tres módulos principales: una base de datos MySQL, un servicio web llamado DOMServer al que se conectan los usuarios para realizar los envíos (*véase Figura 19*) y un proceso externo encargado de realizar encuesta sobre dichos envíos, procesarlos y enviar un *judgement* o veredicto de vuelta a los usuarios. El servicio *DOMServer* cuenta con una API [7] REST que nos va a permitir obtener datos del concurso.

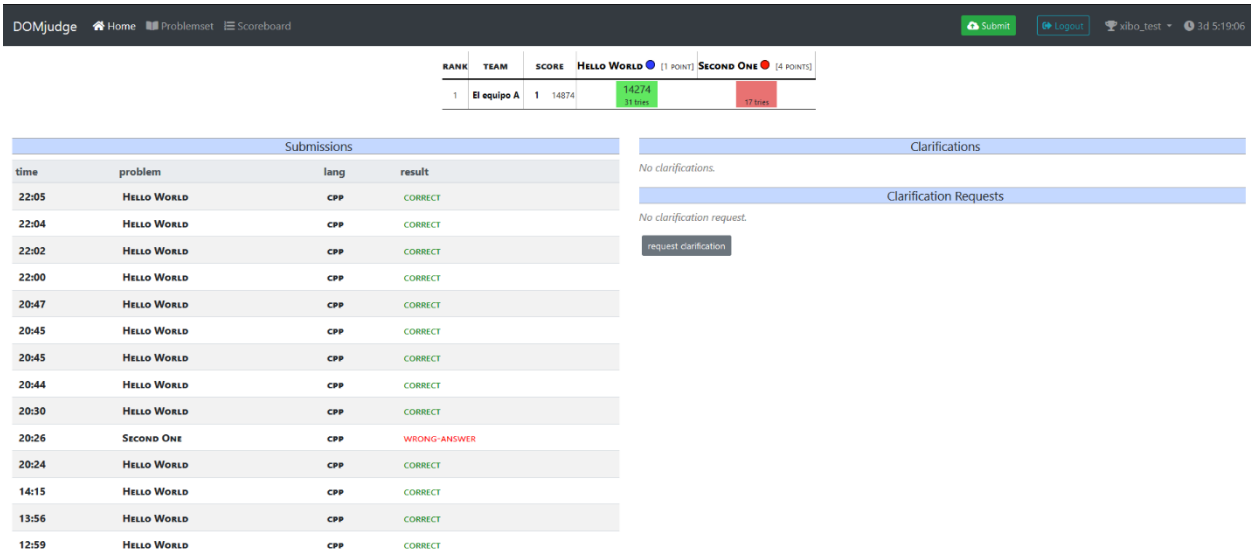

Figura 19. DOMJudge: Pantalla principal

6.2. Arquitectura

Esta aplicación debe integrar, por lo tanto, dos sistemas diferentes para gestionar el contenido a mostrar en las pantallas. Estos sistemas son: DOMJudge y Xibo. A ambos accedemos mediante sus correspondiente API REST. El acceso a la API de Xibo está detalladamente descrito en el capítulo 5 de este documento.

La API de DOMJudge nos proporcionará el estado del concurso en tiempo real y, en función de ese estado, la aplicación hará uso de la API de Xibo para programar el contenido, explicado en detalle más abajo, en las pantallas.

Además, como hemos introducido en la sección 6.1, se pretende mostrar *streaming* de vídeo procedente de un conjunto de webcams situadas en cada ordenador utilizado en el concurso. Este *streaming* se obtendrá con la ayuda de una aplicación adicional, la cual se explica con más detalle en secciones posteriores.

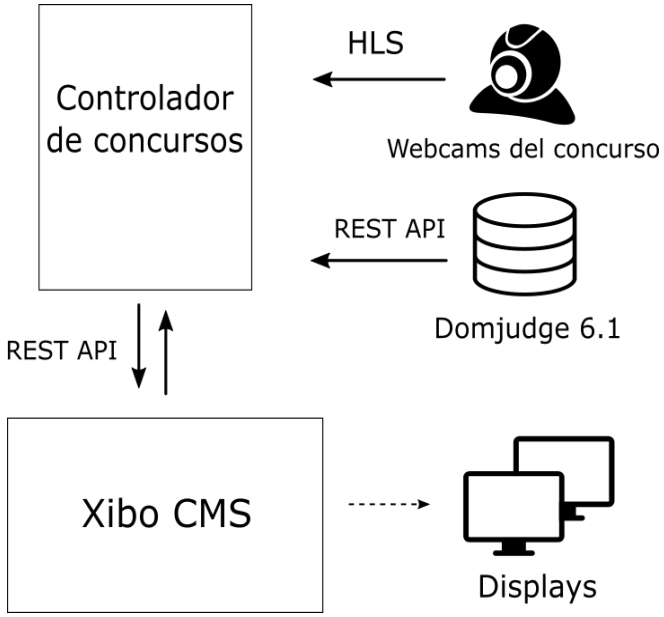

Figura 20. Arquitectura Controlador de Concursos

De esa forma, la arquitectura del nuestro controlador de concursos quedaría de la forma en la que se recoge en la *figura 20*.

En las siguientes secciones del capítulo se va a explicar más en profundidad el alcance y la parte explotada de la API de DomJudge, el servicio de *streaming* necesario para poder mostrar vídeo procedente de las webcams y el contenido relativo al concurso que será mostrado en las pantallas.

6.3. API de DOMjudge

Como ya hemos mencionado, de la API obtendremos información de las diferentes entidades de datos con las que trabaja DOMjudge: *contest* (concurso), *submissions*

(envíos), *judgements* (veredictos), *scoreboard* (puntuaciones) y *teams* (equipos), entre otras.

Existe en la API un método que nos permite obtener acceso a todos los eventos que están ocurriendo en el concurso. En función de dichos eventos sabremos cuando un equipo ha realizado un envío para un problema y su resultado.

6.4. Servicio de streaming

Una de las limitaciones que encontramos en Xibo fue la ausencia de versatilidad a la hora de reproducir streaming de vídeo. Siendo HLS[19] la única opción viable, soportada y recomendada para la versión Windows del reproductor de Xibo.

HLS o *HTTP Live Streaming* es un protocolo de transferencia de vídeo en tiempo real a través de HTTP. Fue creado por Apple en 2009, y es utilizado por la empresa californiana como estándar de streaming de video en la mayoría de sus dispositivos.

El formato HLS funciona de la siguiente manera:

- La señal de vídeo y audio se divide en fragmentos con una duración determinada.
- Existe un archivo que funciona como una lista de reproducción cuya extensión es `.m3u8`. En esta lista de reproducción se especifica el orden de los fragmentos de video y su duración. Es lo primero que se envía al cliente, allí se lee y se empiezan a solicitar los fragmentos al servidor el cual los va enviando en el orden correcto. Toda esta transmisión de datos se realiza por HTTP.

Dado que el formato de video procedente de las webcams no puede reproducirse en el reproductor de Xibo, necesitamos realizar una conversión del formato *raw* o *bruto* de vídeo a HLS. El nombre que recibe una conversión de este estilo es *transcodificación*. Para realizarla, hemos implementado una aplicación compuesta de módulos diferenctes:

[19] Http Live Streaming: https://en.wikipedia.org/wiki/HTTP_Live_Streaming

1. Un transcodificador de video utilizando la biblioteca FFMPEG[20] de tratamiento de vídeo/audio. La señal de video procedente de la webcam conectada a cada ordenador se transcodifica en HLS.
2. Un servicio HTTP mediante el cual poder "servir" el *streaming* de video previamente codificado. Para este servidor se ha utilizado Node.js, implementando el servicio bajo el protocolo de Autenticación básico de HTTP, con el fin de proteger datos de carácter personal de los participantes del concurso.

6.4.1 Manual de instalación

Prerrequisitos:

- Node.js (v10.15.3-LTS at 20/05/19): https://nodejs.org/es/
- NPM (funcionando en v6.4.1): https://www.npmjs.com/
- Cada ordenador funcionando con SO Linux y la webcam en el dispositivo "`/dev/video0`". Todos los ordenadores deben de encontrarse en la misma red que el Controlador de Gestión diaria y el CMS de Xibo.

Para instalar y configurarlo el servidor de streaming correctamente se debe seguir el siguiente procedimiento:

1. Descargar el código fuente desde nuestro repositorio de GitHub: https://github.com/TfgDigitalSignage/HLS-Streaming-Server
2. Una vez tenemos el código descargado ejecutamos el comando `npm i` para descargar las dependencias del proyecto.
3. Procedemos a configurar el fichero "`config/commons.js`". Los únicos parámetros que necesitamos configurar son el usuario y la contraseña de acceso al servidor HTTP. El resto se pueden dejar con su valor predeterminado.
4. Por último, el día del concurso, antes de que comience, lanzaríamos el servicio de streaming cada ordenador deseado con la orden `npm start` sobre la raíz del proyecto.

[20] FFMPEG Free Software Library: https://ffmpeg.org/

6.5. Contenido de las pantallas

En función del momento en el que se encuentre el desarrollo del concurso y de los eventos recibidos de la API de DOMjudge, el controlador de concursos hará visible cierto contenido en las pantallas. Para esto, renderizará la información recibida desde los servicios de DOMjudge y expondrá una ruta, para que un widget (de tipo *webpage)* de Xibo lo consuma.

Además, en todo momento se mostrará un widget de tipo Twitter en el que se recogerán los tweets (que no contengan contenido multimedia) de la cuenta oficial de ProgramaMe[21].

6.5.1 Antes del concurso

En una primera instancia, al iniciar el controlador, se creará una diapositiva que mostrará cíclicamente información general del concurso:

- Información variada sobre el concurso: con el nombre de la competición, su hora de inicio y fin, los distintos problemas de los que va a estar compuesto el concurso (*véase Figura 21*), e información de cada equipo que concursa como el nombre del equipo y el de sus integrantes (*véase Figura 22*).

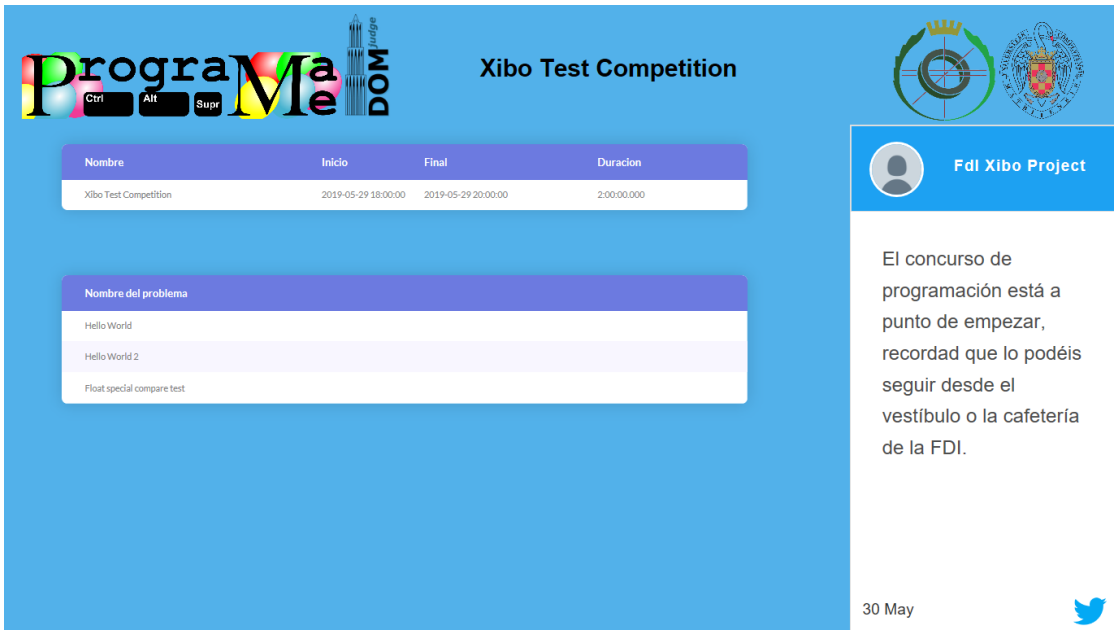

Figura 21. Controlador de Concursos: Pantalla de información del concurso

[21] Twitter de ProgramaMe: https://twitter.com/programame

Figura 22. Controlador de Concursos: Pantalla de información sobre participantes

- Una pantalla con la cuenta atrás para el inicio del concurso (*véase Figura 23*).

Figura 23. Controlador de Concursos: Pantalla de cuenta atrás para el inicio

6.5.2 Durante el concurso

Una vez el concurso haya empezado, la diapositiva verá modificado su contenido para ofrecer otras informaciones sobre cómo va variando el desarrollo del concurso, todas ellas en tiempo real:

- Cuando un equipo realice un envío la programación varía durante unos segundos (parámetro configurable, ver más abajo en la sección manual de instalación). En ese momento se obtiene la imagen de la webcam situada en el ordenador del equipo junto un *feed* de veredicto del envío de modo que podremos ver la reacción del equipo al conocer el veredicto. (*véase Figura 24*).

Figura 24. Controlador de Concursos: Pantalla estado de envío

- Cuando se haya sobrepasado un número dado de envíos, se muestra cada cierto tiempo (periodo de tiempo también configurable), una gráfica con la relación de Envíos/Aciertos para cada problema (*véase Figura 25*). Para elaborar estos gráficos hemos utilizado Google charts [8], por lo que si en algún momento se quisiera se podría cambiar el tipo de gráficos o colores a mostrar (los colores por defecto de Google Chart son el azul y el rojo) desde el fichero `submissions-graphic.pug` en el directorio `view/competition` del proyecto.

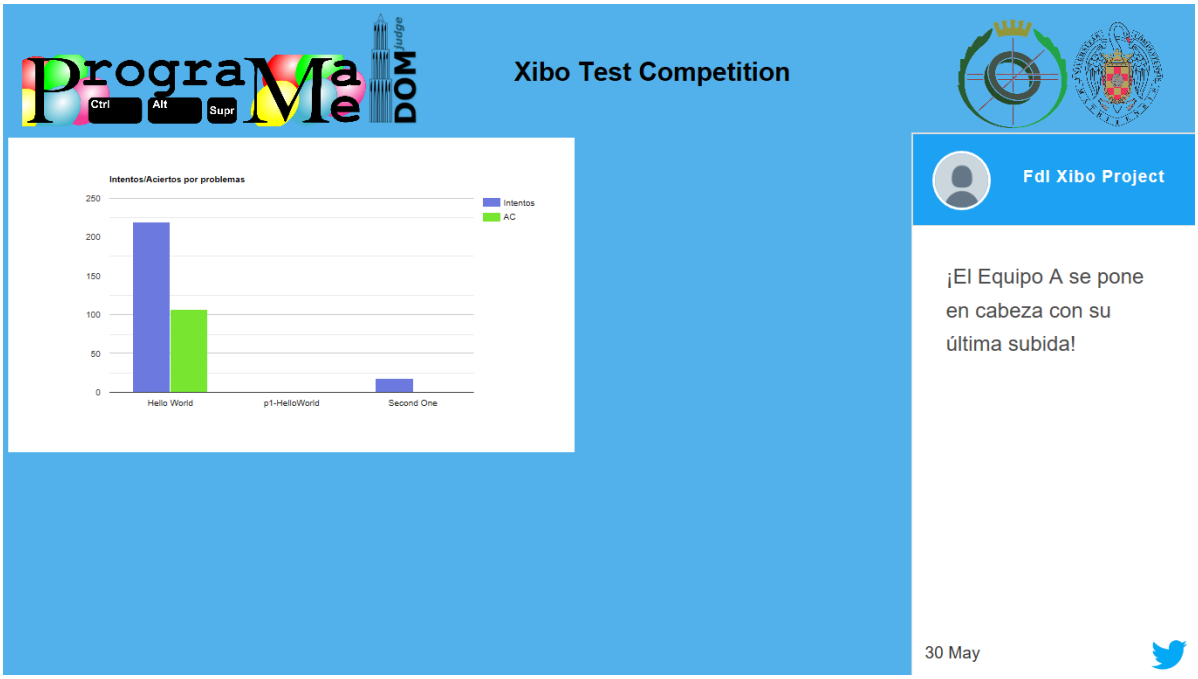

Figura 25. Controlador de Concursos: Pantalla grafica de envios/aciertos

- La clasificación de la competición obtenida en tiempo real de los equipos (*véase Figura 26*). Mostrando el número de problemas resueltos y el tiempo de ejecución total de los problemas enviados.

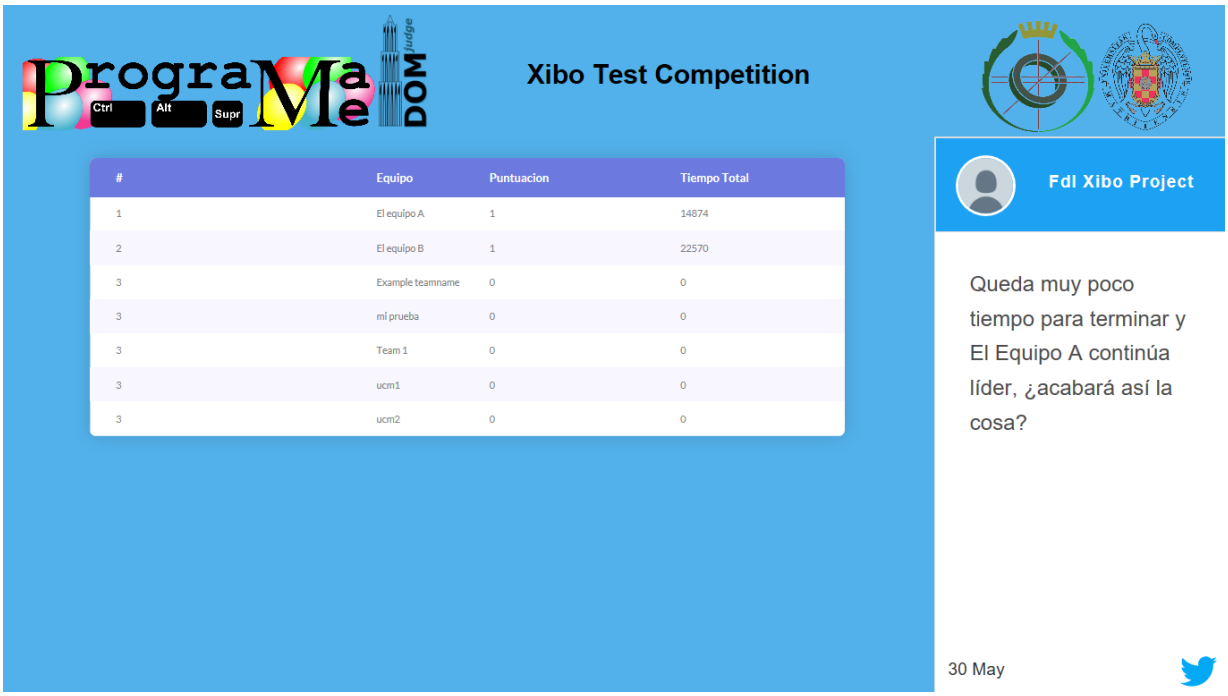

Figura 26. Controlador de Concursos: Pantalla de scoreboard

- Una pantalla con la cuenta atrás para el final del concurso (*véase Figura 27*).

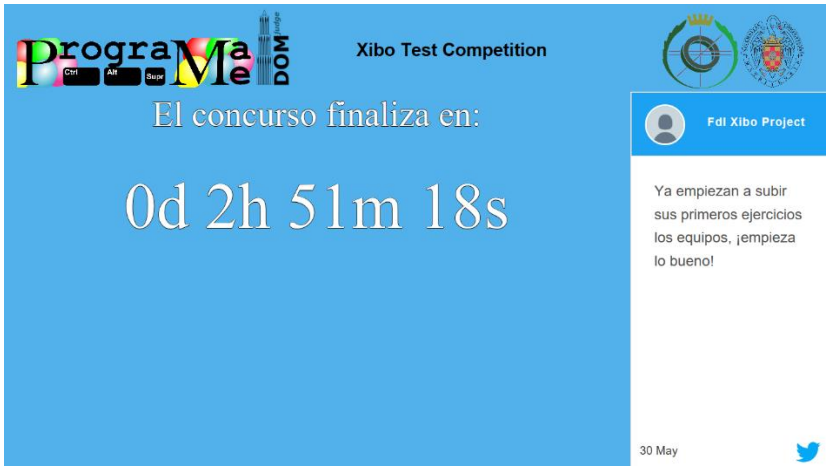

Figura 27. Controlador de Concursos: Pantalla de cuenta atrás para finalizar

6.5.3 Después del concurso

Al finalizar el concurso, se mostrará a modo de resumen información relativa a su resultado:

- La clasificación final de la competición (*véase Figura 26*).
- Una felicitación al equipo ganador que muestre su puntuación entre otros parámetros, como el nombre del grupo, su ID, y los nombres de sus miembros (*véase Figura 28*).

Figura 28. Controlador de Concursos: Pantalla de felicitación

6.6. Despliegue e instalación

La instalación de este Controlador de Concursos está ligada a la del Controlador de Gestión Diaria, por lo que aquí explicaremos cómo configurar esta parte y para ello es necesario haber instalado y configurado correctamente Xibo *(ver sección 4.4. Guía de instalación de Xibo)* y el Controlador de gestión diaria *(ver sección 5.4. Despliegue e instalación).*

6.6.1. Manual de configuración

Esta parte corresponde a la configuración necesaria para que funcione el módulo de concursos como es debido.

Para configurar la función correctamente habrá que, como se hizo anteriormente para el Controlador de Gestión Diaria, darle los valores adecuados al fichero "`.env`". Hay dos secciones de dicho fichero que están involucradas en este controlador de concursos:

1. Parámetros de DOMjudge (*véase Figura 29*):

```
#DomJudge Options
#-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------
## SET USERNAME/PASSWORD WITH REST API READER/WRITTER PERMISSIONS FROM YOUR DOMSERVER.
## API ENDOPOINT MUST END ON '/api/v4/'
## CONTEST_ID IN DOMJUDGE SYSTEM. LOOK FOR YOURS ON YOUR DOMJUDGE JURY INTERFACE: Before Contest Pane >
Contests > All Avaliable Contests (bottom of page)

DOMJUDGE_USERNAME=
DOMJUDGE_PASSWORD=
DOMJUDGE_API_URL=
DOMJUDGE_CONTEST_ID=
```

Figura 29. Fichero de configuración: sección DOMjudge

   Para configurar esto, tendremos que habernos creado previamente un usuario en DOMjudge con permisos de tipo *API Reader* y *API Writter*. En las claves `DOMJUDGE_USERNAME` y `DOMJUDGE_PASSWORD` asignaremos el nombre de usuario y contraseña del susodicho.

   También es necesario escribir la dirección del api de nuestro servidor de DOMjudge en la clave `DOMJUDGE_API_URL` cuya terminación debe ser "`/api/v4/`".

   Por último y no menos importante, necesitamos saber el ID de la competición a utilizar en DOMjudge. Para comprobar el ID de una competición o *contest* accedemos al panel de administración > Contests > All avaliable contests. Encontraremos una lista con los concursos creados y sus IDs.

2. Visualización de gráficos (*véase Figura 30*):

```
## FRECUENCY IN MS OF SUBMISSIONS GRAPHIC SCREEN (default 10 min)
## DURATION OF WEBCAM/SUBMISSION INFO LAYOUT ON SCHEDULE (default and recommended 20secs)
GRAPHICS_SCREEN_FRECUENCY=600000
WEBCAM_VIEW_DURATION=20000
```

Figura 30. Fichero configuración: sección de gráficos

   Para definir cuándo tienen que aparecer los gráficos que muestran el recuento de subidas y aciertos al concurso existe el parámetro `GRAPHICS_SCREEN_FREQUENCY` (por defecto su valor es de 10 minutos, en milisegundos). Por otra parte, el parámetro `WEBCAM_VIEW_DURATION marca` el tiempo en que la webcam de un equipo va a ser mostrada cuando realizan un envío.

```
#HLS Video Server Specifications
#-------------------------------------------------------------------------------------------------------------
-----------------------------------------------
## SET USERNAME/PASSWORD PREVIOUSLY CONFIGURED ON HLS SERVER (only if you're going to use webcam feature on
contests)

HLS_SERVER_USERNAME=
HLS_SERVER_PASSWORD=
```

Figura 31. Fichero de configuración: sección Servidor de Streaming

3. Especificaciones del servidor de streaming:
   En esta sección (*véase Figura 31*), únicamente necesitamos asignar el usuario y contraseña asignados a los servidores de streaming, tal y como se explica en la *sección 6.4* de este mismo capítulo.
4. Configuración del widget de Twitter:
   Para poder hacer uso de este widget es necesario una cuenta de desarrollador de Twitter. Con esto, para configurar el widget hay que acceder al CMS de Xibo y, si no está su módulo instalado, instalarlo desde el panel de Administración en la sección de Módulos, y en la misma sección editar su configuración (incluir clave y secreto de su API, para lo, además de marcar el periodo de caché a 1).
5. Preparación de la diapositiva:
   Para mantener el diseño de diapositiva que utilizamos, hemos añadido un fichero ZIP en la ruta "`/util`". En la sección Diapositivas existe la opción Importar, la cual hay que utilizar para añadir el ZIP en cuestión y, una vez subido, ya tendremos una diapositiva creada con ese mismo diseño. Ahora solo hay que Diseñar esta diapositiva y desde el panel de diseño utilizar la opción Guardar como plantilla, con el nombre de "ContestLayoutTemplate", para que posteriormente nuestra herramienta la reconozca y pueda trabajar con ella.

6.6.2. Posible ampliación

En el caso de que, en un futuro, se quisiera ampliar la funcionalidad de esta aplicación, recogemos en este apartado una pequeña guía con recomendaciones para hacerlo de la forma más sencilla posible.

Esta aplicación, como ya hemos explicado más arriba, hace uso de dos sistemas externos que son Xibo y DOMjudge. En el directorio `services` de nuestro proyecto, se encuentran los dos ficheros en los que se realiza la implementación de un cliente para cada

una de los dos APIs. Como se explica en la sección 5.4.2 de este documento, *acerca de la ampliación del controlador diaria,* no hemos aprovechado toda la funcionalidad que nos ofrece una herramienta como Xibo, por lo que si en un futuro, fuera necesario, se podría ampliar dicha funcionalidad.

Lo mismo ocurre para DOMjudge. Se ha hecho un análisis sobre la información requerida para esta primera versión, pero se puede seguir explotando el potencial de su API en el futuro. De hecho, nosotros animamos a hacerlo. Por ello, hemos organizado el código de manera que pueda ser lo más escalable posible, para permitir la ampliación de funcionalidad.

Una vez implementados métodos en el fichero `services`, la información proveniente de las diferentes APIs (usualmente en formato JSON[22]) es procesada por los métodos en los ficheros del directorio `controller`. Dependiendo del tipo de dato a procesar, usaremos un controlador u otro.

Por último, en el directorio `routes` se encuentran los ficheros encargados del enrutamiento de las peticiones realizadas desde el navegador (por parte del administrador) o desde los widgets de Xibo (que serán programados para visualizarse en las pantallas). De entre estos, el fichero "`competition.js`" gestiona las rutas relativas al concurso, y de entre estas cabe destacar que en la función que maneja la ruta "`/start`" se reconocen los eventos que tienen lugar en el concurso, como las subidas o los veredictos, y ahí se podrían manejar más eventos que ofrezca el concurso en el caso de querer ampliarlo.

Estas "rutas" renderizan el contenido en HTML5 utilizando *pug* como motor de plantillas. Los distintos ficheros "*.pug"* podemos encontrarlos en el directorio `views`. Estos ficheros son los que se deberían modificar en el caso de querer cambiar algo de la parte gráfica que ofrece información sobre el concurso, como son la tabla de clasificación, el tiempo restante, sus colores, etc.

Por otra parte, para modificar la parte gráfica que muestra el controlador de concursos pero que no trata del concurso, como son la cabecera con sus logos y la parte de Twitter, hay que modificar los widgets incluidos en la diapositiva a utilizar. La cabecera consiste en un widget de tipo Embedded en el que los logos se encuentran en código HTML y CSS, y el nombre del concurso se añade mediante código JavaScript. Por otra parte, el widget de Twitter se puede modificar también para cambiar la fuente del contenido a mostrar, el tipo de plantilla a utilizar, el número de tweets, su duración, etc.

---

[22] JSON: *JavaScript Object Notation: https://es.wikipedia.org/wiki/JSON*

Capítulo 7. Conclusiones

Hoy en día la publicidad es un sector que mueve mucho dinero, además de un área crítica para las empresas. Además de esto, el posicionamiento estratégico de esta publicidad es clave para llegar a cuanta más gente mejor. Por ello, la cartelería digital ha cobrado mucho interés y relevancia, y en la actualidad podemos encontrar en muchos sitios estos carteles digitales en los que se sucede publicidad en las pantallas.

En lo que respecta a nuestra facultad y nuestro proyecto, si sustituimos publicidad por información, hacer uso de esta técnica puede despertar el interés y la curiosidad de los estudiantes; y en cuanto al posicionamiento de las pantallas existentes, la ubicación que tienen es perfecta para llegar a la mayor parte de los estudiantes.

Con esta base, el aprovechamiento de estas pantallas por parte de la facultad es todo un acierto, pero al sistema actual le hacían falta algunas mejoras para ser capaz de ofrecer una información más dinámica y hacer posible mostrar contenido en tiempo real. Este proyecto cubre esta necesidad, haciendo uso del potencial de la herramienta de Xibo, como demostramos con nuestra prueba de concepto.

Además, con nuestro Controlador de Gestión Diaria, mantenemos la sencillez de la gestión del contenido de las pantallas. El usuario tan solo necesitará acceder a nuestro controlador para programar el/los eventos/s para ese día, semana, o para el tiempo que desee con tan solo unos pocos pasos.

Todo esto será posible tras la instalación de nuestro sistema, compuesto de tres herramientas diferentes, y unas pequeñas configuraciones.

7.1 Posibles mejoras futuras

Como posible mejora, además de las modificaciones o adiciones explicadas anteriormente, tenemos la idea de que en un futuro se podría realizar una app personalizada para móvil de nuestro Controlador de Gestión Diaria, de modo que el usuario tenga a su disposición una aplicación que le facilite la gestión del contenido de las pantallas con la máxima comodidad.

7.1.1. Mejoras para el Controlador de Gestión Diaria

Nuestro controlador no cubre todas las opciones que ofrece la API de Xibo, y en este apartado vamos a incluir algunas de las posibles modificaciones más importantes que se podrían hacer sobre nuestro código:

- Layout Manager:
  - Gestión de regiones de una diapositiva:

    La API de Xibo cuenta con métodos `POST`, `PUT` y `DELETE` sobre las regiones de una diapositiva que no hemos utilizado. Para trabajar con ellos bastaría con añadir las funciones que realicen estas peticiones a la API en el fichero "`services/xiboServices.js`", y llegar hasta estos métodos siguiendo la estructura de nuestra aplicación.
  - Más opciones en la creación de widgets:

    En nuestra implementación de creación de widgets para añadirlos a una diapositiva añadimos los parámetros necesarios para la creación de estos widgets, pero según el tipo de widget hay otros muchos parámetros opcionales que pueden añadirse y que no abarcamos.

    Para añadir estos parámetros bastaría con editar tres ficheros: la función `addWidget` de "`controller/layoutController.js`", la función correspondiente de "`services/xiboServices.js`" y la vista "`views/addWidget.pug`".

- Campaign Manager:
    - Eliminar una diapositiva de una campaña:

      Esta funcionalidad podría aparecer con un botón de eliminación que obtenga la información de un formulario que englobe la lista de diapositivas existentes en la campaña, siendo cada uno de estos elementos el contenido de un tipo de `input` "`radio`". Para implementarla, se podría realizar una simetría con la misma opción existente en el Layout Manager que elimina un widget de una diapositiva, y hacerlo de la misma forma.

- Event Manager:
    - Editar un evento:

      La API de Xibo cuenta con un método `PUT` para editar eventos, el cual no hemos utilizado. Para diseñar la edición de eventos bastaría con editar la vista añadiendo un formulario a la lista de eventos existentes y generar una nueva con las opciones de edición del evento. En cuanto a la implementación, habría que crear un nuevo método en el fichero "`services/xiboServices.js`", y en los distintos ficheros correspondientes a eventos que se encuentran en la estructura de nuestra aplicación.

    - Recibir lista de eventos existentes programados para otro día

      Actualmente nuestro controlador muestra los eventos programados en el momento en que se accede. Para que este comportamiento cambie, habría que modificar la función `getEvent` del fichero "`controller/EventController.js`", donde indicamos que coja el `dateTime` actual y se lo mande al CMS en la petición. Si en lugar de esta fecha y hora, en la función invocada debajo, que conecta con el CMS, le pasamos otra como puede ser la de mañana (bien desde ahí, bien desde la vista mediante la interacción con el usuario).

7.2 Contribuciones

En este punto hablamos en primera persona de las contribuciones de cada uno de los componentes del grupo al proyecto, incluyendo los diferentes aprendizajes y prácticas que hemos llevado a cabo en su desarrollo, hayan o no estado involucrados directamente en la versión final del proyecto.

7.2.1. Contribuciones de Adrián Montero Torralbo

- Participación en el proceso de investigación sobre diferentes herramientas de dashboard y digital signage:
  Tanto yo como mis compañeros realizamos primeramente una investigación acerca del estado del arte. En este proceso aprendí el concepto de dashboard, sobre el que probé el software Freeboard; por otra parte, me topé con el concepto de digital signage, el cual me pareció el más acertado para el desarrollo de nuestro proyecto tras aprender sobre él. En cuanto a software de digital signage, indagué sobre las estructuras y el funcionamiento de las herramientas privativas Mural Digital y Smush Digital, y probé brevemente la herramienta de software libre Concerto, antes de probar y centrarnos en la herramienta de Xibo. (Todas las herramientas mencionadas anteriormente se encuentran recogidas en el capítulo 2 de este documento).

- Instalación e investigación de Xibo:
  Desde el momento en que descubrimos la herramienta de Xibo, he participado en su investigación, al igual que mis compañeros, hasta entender y dominar casi la totalidad de las funcionalidades que ofrece. De entre estos, los aspectos más influyentes para nuestro proyecto han sido las pruebas con diapositivas y campañas, usando casi todos los tipos de widgets disponibles (sobre todo Text, Image, Video, Webpage, Embedded, Calendar, Twitter), y su posterior programación en eventos. Para esto último, he probado y medido tiempos de funcionamiento en el reproductor de Windows y en la versión en desarrollo del reproductor de Linux.

Además, investigué la creación de nuevos módulos (que dan lugar a nuevos tipos de widgets), hasta llegar a desarrollar dos de ellos, los cuales no se han utilizado finalmente en el proyecto.

- Controlador de gestión diaria:
  En esta parte, que la hemos hecho entre los tres miembros del grupo, he participado sobre todo en el desarrollo final de la misma, y no tanto en la fase inicial.
  En un primer momento empezamos cogiendo el contenido a mostrar de una lista de contenidos en un fichero JSON estático. Posteriormente hicimos que este fichero pudiera ser modificado y nuestro controlador fuese capaz de detectar estos cambios y modificar el contenido a mostrar; en esta parte me encargué de realizar un método efectivo de encuesta sobre este fichero y su posterior ejecución.
  Finalmente, cuando el funcionamiento de la aplicación era más parecido al actual, participé, tanto en la parte gráfica como en su desarrollo interno, en la creación y eliminación de eventos, diapositivas (y sus widgets) y campañas, y en la edición de estas dos últimas entidades. También me he encargado de la obtención y visualización de listas de diapositivas, campañas y eventos actuales.

- Controlador de concursos:
  Para el controlador de concursos, como en el apartado anterior, mi aportación ha tenido lugar sobre todo en su fase final de desarrollo.
  Me he encargado de realizar los diseños y la implementación en el código de las pantallas que muestran una cuenta atrás (antes y durante el evento) y la pantalla final de felicitación al ganador. En cuanto al desarrollo del funcionamiento de la aplicación durante el concurso, me he encargado de la gestión a la hora de reconocer y tratar un nuevo envío al juez y su posterior veredicto, participando también en el desarrollo inicial de la pantalla que muestra su información.

- Contribuciones a la memoria:
  En cuanto a este documento, mis compañeros y yo hemos participado finalmente en todos los capítulos.

Mi contribución inicial han sido principalmente el resumen, los capítulos 1 (Introducción), 3 (Estado del arte), 4 (Xibo), y 7 (Conclusiones).
Y finalmente, como al principio de este punto, he participado en las correcciones de todos los capítulos existentes en el proyecto, además de en la elaboración de los capítulos 2 (Introduction) y 8 (Conclusions).

7.2.2. Contribuciones de Gabriel Gutiérrez Santos

- Participación en el proceso de investigación sobre diferentes herramientas de dashboard y digital signage:
  Mi contribución a esta parte ha sido la misma que la del resto de mis compañeros. Investigación sobre el concepto de dashboard y digital signage, y pruebas con diferentes herramientas de un tipo y del otro.

- Instalación e investigación de Xibo:
  En esta parte, he contribuido en la investigación inicial de Xibo y las funcionalidades y aspectos que cubría, dentro del digital signage (como mis compañeros). Ante todo, a lo que más tiempo he dedicado a esta tarea (al principio del proyecto) es a la investigación sobre la API de Xibo: Uso, corrección de errores, etc.

- Controlador de gestión diaria:
  En esta parte, que la hemos hecho entre los tres miembros del grupo, participé más activamente al comienzo que al final.
  Al principio del todo, partimos de los conocimientos básicos acerca de la API de Xibo para implementar un pequeño script que obtuviera datos de un JSON y en función de estos, cambiara el contenido en el CMS.
  Posteriormente trasladamos el desarrollo a node.js y seguí trabajando en esta parte hasta que surgió la necesidad de crear un Servidor de Streaming y el controlador del concurso. Estas dos partes, son en las que más tiempo he invertido.

- Servidor de Streaming:

  Para esta parte, comencé con una investigación de los diferentes formatos de video soportados por Xibo. Probamos con varios de ellos antes de decantarnos por HLS. A partir de aquí empecé a investigar diferentes servidores que ofrecieran una implementación HLS y escogimos hacerlo en Node.js dado que ya lo habíamos estado utilizando para el resto de la aplicación. También me atribuyo la investigación de diferentes *módulos* de node.js que facilitaran la tarea y de las librerías de video, de las cuales se escogió FFMPEG. Finalmente implementé el servicio de Streaming tal y como se indica en el apartado 6.4.

- Controlador de concursos:

  Me he encargado de implementar toda la lógica de decisión del contenido a mostrar durante los concursos (en base a ideas de mis compañeros y mi tutor). También investigué e implemente los primeros métodos que hacían uso de la API de DomJudge. Respecto al contenido a mostrar, he hecho la pantalla de la clasificación del concusro y terminé de implementar la gráfica de envíos/correctos por problema; escogí Google Charts para dibujarlas, optimicé el uso de peticiones a la API para que no se quedara bloqueado el servicio de DOMJudge e implementé la pantalla del veredicto en directo de un envío.

- Diagramas de arquitectura de los distintos sistemas: Creación de los diagramas vectoriales del Sistema actual, el sistema mejorado y el controlador de concursos. Todos ellos han sido creados con la herramienta de GNU InkScape.

- Contribuciones a la memoria:

  En cuanto a este documento, mis compañeros y yo hemos participado finalmente en todos los capítulos.

  Mi contribución inicial han sido principalmente los capítulos: 5 (Controlador de gestión diaria) y 6(Controlador de concursos).

  Y finalmente, como al principio de este punto, he participado en las correcciones de todos los capítulos existentes en el proyecto.

7.2.3. Contribuciones Daniel Gutiérrez Delgado

- Participación en el proceso de investigación sobre diferentes herramientas de dashboard y digital signage:
  Por mi parte en este apartado también realicé el estudio de los dos conceptos ya mencionadas anteriormente. Realizando una compresión del concepto de dashboard y buscando herramientas de este tipo en internet y que hemos explicado en el apartado 3.2 de esta memoria. Realicé el mismo proceso para digital signage, y que finalmente fue la opción que decidimos tomar para elaborar este trabajo y como para el dashboard busqué herramientas para tener una batería de pruebas de la cual posteriormente seleccionamos la mencionada anteriormente en el apartado 4 de esta memoria.

- Instalación e investigación de Xibo:
  Una vez seleccionada la herramienta que más se ajustaba a nuestras necesidades procedimos a realizar un estudio completo de la misma para poder conocer por completo todas las funcionalidades que nos ofrecía (programación de eventos, widgets disponibles e instalación de otros ya reconocidos, pero no instalados, creación y edición de layouts) y cómo optimizar la herramienta al máximo para así poder ajustarla por completo a desarrollar este trabajo, Dentro de esta parte de optimización cabe destacar el XMR, del cual realicé un estudio y aplicamos posteriormente a nuestro proyecto para una optimización en el tiempo de comunicación entre el CMS y el reproductor. También realicé una investigación sobre cómo introducir módulos propios en Xibo, consiguiendo instalar un módulo creado por mí mismo en las rutas correctas y sin ninguna funcionalidad para poder averiguar cuáles eran las funciones mínimas necesarias para que el CMS reconociera el módulo y lo instalara correctamente.

- Controlador de gestión diaria:
  Como ya se ha mencionado anteriormente esta parte del trabajo fue realizada por todo el equipo.
  De esta parte yo me centré más en la parte de los eventos ayudando en su creación, en recoger todos los eventos que habían sido creados para mostrar posteriormente

la información de los eventos programados para el día y con dicha información eliminar el evento. Al igual hice con la parte de las campañas, creando una campaña vacía, para después agrupar layouts seleccionados por el usuario e introducirlas en esa campaña creada y mostrar la campaña con su información para poder eliminarla cuando se quisiera.

- Controlador de concursos:
  En este apartado yo me he encargado de realizar la implementación en el código de las pantallas que muestran los gráficos con los envíos realizados y el número de aciertos por cada problema, y para antes del concurso me encargué de recoger la información del concurso como el nombre del concurso, las fechas de hora de inicio y de fin del concurso y el tiempo de la duración del concurso. Por otro lado, también se muestra la información de los equipos que van a participar, mostrando tanto el nombre del equipo como el de cada uno de los participantes de dicho equipo, y por último se mostrarán los nombres de los ejercicios que habrá en el concurso.

- Contribuciones a la memoria:
  En la redacción de este documento, como se ha comentado anteriormente, se ha realizado de manera conjunta por todos los integrantes del equipo.
  Por mi parte me he centrado más en los capítulos 5 (Controlador de gestión diaria), y 4 (Xibo).
  También junto con el resto del equipo he realizado correcciones de los errores en todos los capítulos de esta memoria.

7.3. Recursos utilizados

Aquí exponemos un listado con las diferentes tecnologías empleadas en el desarrollo de nuestro proyecto.

- Sistema operativo: Windows 10 para el reproductor de Xibo; Ubuntu 18.0.4 para el CMS y el desarrollo y ejecución de nuestra aplicación.

- Navegador web: Chrome, Chromium y Firefox para probar el correcto funcionamiento de nuestra aplicación, y también del CMS de Xibo, en diferentes navegadores.
- Control de versiones de la aplicación: Git para el desarrollo en local, facilitando el trabajo en paralelo, vinculado a un repositorio GitHub.
- Desarrollo del código: sobre el editor de textos SublimeText y Visual Code.
- Servidor: basado en Node.js sobre una infraestructura Express.js.
- Lenguajes de programación:
    - JavaScript: en el código de nuestra aplicación y AJAX, utilizado para algunas comunicaciones entre nuestra parte gráfica y DOMjudge en el Controlador de Concursos.
    - HTML: en dos formatos diferentes, PUG en nuestro servidor (motor de plantillas de Node.js), y TWIG en la creación y modificación de módulos de Xibo (motor de plantillas de PHP). Además, hemos añadido algún CSS utilizando Bootstrap.
    - PHP: en la creación y modificación de módulos de Xibo.
- Diseño de imágenes vectoriales: con InkScape, para crear diagramas mostrados en este documento.

Capítulo 8. Conclusions

Nowadays advertising is a sector that moves loto f money, moreover it is a critical area for companies. In addition to this, strategic positioning of this advertising is key to reach as the more many people better. For this reason, digital signage has gained lot of interest and relevance, and currently we can find in many places these digital posters in which advertising is changing itself on screens.

Regarding our faculty and our project, if we substitute advertise for information, making use of this technique can arouse students interest and curiosity; and regarding the existing screens positioning, the location they have is perfect to reach most of the students.

With this base, exploitation of these screens by the faculty is a success, but current system needed some improvements in order to be able to offer a more dynamic information and make possible to display real time content. This project cover this need, making use of the Xibo tool potential, as we demonstrate with our proof of concept, Contests Controller.

In addition, with our Daily Management Controller, we keep the simplicity of content management of screens. User will only need to access our controller in order to program the event/s for that day, week, or for the time he want with just a few steps.

All this will be possible after the installation of our system, consisting of three different tools, and just a few configurations.

Bibliografía

[1] NeoAttack. (2018, 16 octubre). ¿Qué es un Dashboard y para qué sirve? Recuperado 24 mayo, 2019, de https://neoattack.com/neowiki/dashboard/

[2] MarTech Forum. (2019, 11 febrero). Qué es la Señalización digital (Digital Signage). Recuperado 25 mayo, 2019, de https://martechforum.com/articulo/que-es-digital-signage/

[3] India, C. (2017, 26 abril). ¿Qué es Digital Signage? La revolución de la publicidad digital. Recuperado 25 mayo, 2019, de https://www.cyberclick.es/numerical-blog/que-es-digital-signage-la-revolucion-de-la-publicidad-digital

[4] Trisolino, D., & Cappellacci, M. (2018, 30 enero). Testimonial from Università degli di Urbino Carlo Bo. Recuperado 23 mayo, 2019, de https://blog.xibo.org.uk/testimonial-universita-degli-di-urbino-carlo-bo-italy/

[5] Arnold, J. (2017, 11 diciembre). Testimonial from Longhill High School. Recuperado 23 mayo, 2019, de https://blog.xibo.org.uk/testimonial-longhill-high-school/

[6] Xibo Signage LTD. (s.f.). Xibo CMS API. Recuperado 23 mayo, 2019, de https://xibo.org.uk/manual/api/

[7] DOMjudge. (s.f.). DOMjudge API v4. Recuperado 23 mayo, 2019, de https://www.domjudge.org/demoweb/api/doc

[8] Google. (s.f.). Charts | Google Developers. Recuperado 25 mayo, 2019, de https://developers.google.com/chart/